An Agent-Based Service-Oriented Approach to Evolving Legacy Software Systems into A Pervasive Computing Environment

PhD Thesis

Ruimin Liu

Software Technology Research Laboratory

De Montfort University

2010

To *my husband*, Qing Lin,
my son, Xiuming Lin,
my parents and
parents-in-law.

Declaration

I declare that the work described in this thesis was originally carried out by me during the period of registration for the degree of Doctor of Philosophy at De Montfort University, U.K., from April 2003 to March 2009. It is submitted for the degree of Doctor of philosophy at De Montfort University. Apart from the degree that this thesis is currently applying for, no other academic degree or award was applied for by me based on this work.

Acknowledgements

For many years I had been dreaming about receiving a PhD. I would like to thank many people who helped me in achieving this dream in different ways when I undertook the work of this thesis.

I wish to express my most profound thanks to my supervisors, Prof. Hongji Yang and Prof. Yunsheng Zhang, for their invaluable advice, experienced guidance and encouragement during my four-year study. They provided me with many useful comments and suggestions for preparation of this thesis.

My thanks must go to Professor Hussein Zedan, the director of the laboratory, for his valuable advice and the experienced guidance that he had provided along the way. My research career benefits tremendously from the research methodologies that Prof. Hussein Zedan introduced to me.

I wish to thank Prof. Rachael McCrindle and Prof. Hussein Zedan for examining the thesis carefully and the invaluable comments they gave to me.

I would also like to thank the Research Office at De Montfort University for their outstanding management.

I would like to thank all of the colleagues in Software Technology Research Laboratory at De Montfort University and the colleagues in Kunming University of Science and Technology, for their valuable suggestions and discussions, for their encouragement and support. Many thanks to Feng Chen, for his suggestion, support and concern. Thanks to Zhuopeng Zhang and Shaoyun Li for their suggestion and encouragement. I am indebted to all of them.

Finally, I wish to express thanks to my husband, Qing Lin, my son, Xiuming Lin, my parents and parents in law for all their loves, encouragements, patience and supports over the years. This thesis is dedicated to them.

Abstract

Mark Wesier described his vision of *Ubiquitous Computing* (which now is also called *Pervasive Computing*) in a seminal paper in 1991. His vision is becoming a reality: the ever-increasing availability of inexpensive computation and storage has introduced computers into nearly every facet of our everyday lives, while a revolution in communications has brought high-bandwidth communications into our homes and offices. Wireless communications also have exploded, making digital services available nearly everywhere.

Pervasive Computing will be a fertile source of challenging research problems in computer systems for many years to come. Many research organisations represent a broad communal effort. There are many useful utilities can be used, both in enterprise and in research field, include ubiquitous devices, software for Pervasive Computing environment and infrastructure. But until today, there is less attention to the software evolution in Pervasive Computing environments.

This thesis focuses on an Agent-Based Service-Oriented approach to evolving legacy system into a Pervasive Computing environment. The methodology consists of multiple phases: using reverse engineering techniques to comprehend and decompose legacy systems, employing XML and Web Services to transform and represent a legacy system as pervasive services, and integrating these pervasive services into pervasive computing environments with agent based integration technology.

A legacy intelligent building system is used as a case study for experiments with the approach, which demonstrates that the proposed approach has the ability to evolve legacy systems into pervasive service environments seamlessly. Conclusion is drawn based on analysis and further research directions are also discussed.

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Lists

List of Acronyms

ADL	Architecture Description Language
API	Application Programming Interface
AST	Abstract Syntax Tree
BNF	Backus-Naur Form
EBNF	Extended Backus-Naur Form
EJBs	Enterprise JavaBeans
ER	Entity-Relationship
TAGDUR	Transformation and Automatic Generation of Documentation in UML through Re-engineering
GUI	Graphic User Interface
IDL	Interface Description Language
LOC	Line Of Code
MA	Maintainer's Assistant
MAS	Multi-Agent System
MIS	Management Information System
MOF	Meta-Object Facility
NFR	Non-Functional Requirement
OMG	Object Management Group
RMI	Remote Method Invocation
SERG	Software Evolution Research Group
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
STRL	Software Technology Research Laboratory
TAM	Temporal Agent Model
UDDI	Universal Description Discovery and Integration
UML	Unified Modelling Language

WSL	Wide Spectrum Language
WSDL	Web Service Description Language
WSW	Web Services Wrapper
XMI	XML Meta-data Interchange
XML	eXtensible Markup Language
URI	Uniform Resource Identifier

Chapter 1
Introduction

Objectives

- To motivate the need for evolutionary system development in Pervasive Computing environment.
- To explain the research characteristics and select the research method.
- To identify the research questions and illustrate the research hypothesis.
- To highlight original contribution and define the success criteria.

1.1 Motivation

Mark Weiser described a hypothetical world in which humans and computers were seamlessly united in a seminal article in 1991; he called this vision Ubiquitous Computing [125] (which now is also called Pervasive Computing). After that, technology has advanced along many dimensions, especially in hardware progress and wireless communication technologies. A number of leading technological organisations are exploring Pervasive Computing. But it is far from Weiser's vision become reality.

From mainframes to personal computer, computers are part of everyday life and an inevitable component when performing a variety of private and business related tasks. Beyond the era of personal computing, the era of pervasive computing begins: A new class of devices makes information access and processing easily available for everyone from everywhere at anytime. Users get enabled to exchange and retrieve information

they need quickly, efficiently, and effortlessly, regardless of their physical location.

Pervasive computing is the method of enhancing computer use by making many computers available throughout the physical environment [127], but making them effectively invisible to the user [126]. Pervasive computing names the third wave in computing.

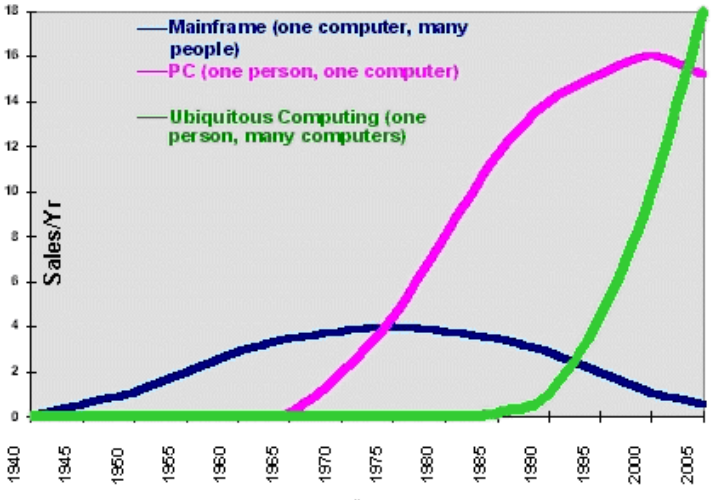


Figure 1-1. Major Trend in Computing [127]

Pervasive computing is fundamentally characterised by the connection of things in the world with computation. Pervasive computing just might help to free the minds from unnecessary work, and connect people to the fundamental challenge that humans have always had: to understand the patterns in the universe and ourselves with them. Weiser heralded a future of ubiquitous and invisible computing in a now often referenced article in Scientific American in which he stated: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." When articulated, this was a vision too far ahead of its time -- the hardware technology needed to achieve it simply did not exist.

A pervasive computing system should be pervasive, embedded, nomadic, adaptable, powerful, yet efficient, intentional and eternal. It must everywhere, with every portal reaching into the same information base; live in our world, sensing and affecting it;

allow users and computations to move around freely, according to their needs; provide flexibility and spontaneity, in response to changes in user requirements and operating conditions; free itself from constraints imposed by bounded hardware resources, addressing instead system constraints imposed by user demands and available power or communication bandwidth; enable people to name services and software objects by intent, as opposed to by address; never shut down or reboot; components may come and go in response to demand, errors, and upgrades [45].

After a decade of technology progress (include new technologies -- such as the Global Positioning System (GPS [22], smart cards, and radio frequency identification (RFID) tags -- and social developments such as the increasingly widespread acceptance of video surveillance in public places), many critical elements of pervasive computing are now viable commercial products, such as: handheld and wearable computers; wireless LANs; and devices to sense and control appliances.

A number of leading technological organisations are exploring Pervasive Computing. Xerox's Palo Alto Research Centre (PARC), for example, has been working on pervasive computing applications since the 1980s. Although new technologies are emerging, the most crucial objective is not, necessarily, to develop new technologies. IBM's project Planet Blue, for example, is largely focused on finding ways to integrate existing technologies with a wireless infrastructure. Carnegie Mellon University's Human Computer Interaction Institute (HCII) is working on similar research in their Project Aura, whose stated goal is "to provide each user with an invisible halo of computing and information services that persists regardless of location." The Massachusetts Institute of Technology (MIT) has a project called Oxygen. MIT named their project after that substance because they envision a future of pervasive computing devices as freely available and easily accessible as oxygen is today.

Pervasive Computing will be the future [44]. Pervasive computing will be a fertile source of challenging research problems in computer systems for many years to come. There are some major research challenges in pervasive computing environment [35], such as software architecture, networking challenge, component interaction, user interface integration, e.g. [36]. But until today, there is less attention to the software

evolution in pervasive computing environment.

Pervasive computing system also has the same characteristics as other computing systems, though it has many more requirements than prior computing system, such as distributed system and mobile computing system. So the development of pervasive computing is also an evolutionary process [93]. Because many research organisations represent a broad communal effort to make pervasive computing a reality, there are many useful utilities can be used, both in enterprise and in research field, include ubiquitous devices (PDAs, smart devices, e.g.), software for pervasive computing environment (Java, operating systems, middleware components, e.g.) and infrastructure (wireless networks…). Based all these contributions, we attempt an evolutionary design approach in pervasive computing environment, which is an Agent-Based Service-Oriented approach to evolving a legacy system into pervasive computing environments. The key process includes component identification from legacy systems, component based service migration and seamless agent-based service integration [101].

In particular, this thesis aims to addressing the following research issues in the area of software evolution towards pervasive computing environment and system integration.

- Legacy system decomposition and components identification. In this part, the program slicing technique is used to decompose system, understand program, eliminate dead code and make selected code segments function independently by component interface parameters determination and deep source code comprehension. The clustering analysis technique is used to group large amounts of entities in a dataset and capture reusable legacy code segments into clusters according to their relationship and similarity from legacy systems, and to create a hierarchical structure of these reusable legacy code segments. The program transformation technique based on Extended Wide Spectrum Language (EWSL) also used in legacy system decomposition and component identification.

- Component based service migration and package. This part presents the approach to migrating and packing the extracted legacy assets as Pervasive. It is based on XML representation and transformation. Once a software component has been

extracted from a legacy system, or has been built as a new component, its interface can be extracted and represented in XML. The representation is finished not only by the component wrap, but also with the sources code analysis, such as the using of AST, DTD and XSLT. At last, the legacy components are wrapped as Pervasive Services which could be further used in the Pervasive Computing environment.

- Agent-based services integration. This part presents a framework that integrates the pervasive services into the Pervasive Computing environment by agent-based techniques. It describes a method for defining the legacy resources as stateful resources, and integrating the pervasive services based on these reusable resources.

This agent-based service-oriented reengineering methodology brings more flexibility, expansibility and reusability, as well as more reliability.

1.2 Research Methods

This section describes the research method applied in this thesis, which links the new knowledge coming from research to the process leading to outcomes. The research field in this thesis belongs to software engineering aiming to be a rigorous discipline and enable the successful production of software (high quality products at the lowest possible cost). Being a kind of computer science and like all kinds of engineering, the majority of software engineering research is constructive. Constructive research refers to the new contributions being developed. A new contribution can be a new theory, algorithm, model, framework or a method. Since software engineering always involves complex action and interaction of human beings, empirical research is also required to investigate such situation. Hence, the research method applied in this thesis is the combination of empirical and constructive research that is of both high practical utility and academically rigorous. The basic methods used in this thesis are summarised as follows:

- Formal methods: Formal methods can be defined as mathematically based

languages, techniques, and tools for specifying and verifying systems. Formal methods can increase the understanding of a system by revealing inconsistencies, ambiguities, and incompleteness that might go undetected [25].

- Quantitative and qualitative methods: Both methods are used under the umbrella of the proposed framework, asking Why, What, How, Who, When, Where questions and looking at the problem and solutions from many points of view.

- Modelling: as means of communication, documentation, analysis, simulation, decision making and verification. Modelling can break up a large problem into smaller problems and reduce the complexity.

- Artificial intelligence: when information needed is still missing, artificial intelligence is required.

A framework for the proposed research method is defined in both science and engineering aspects. The research structure and processes were realised as follows:

Step 1: Research Problem, Question and Hypothesis Identification.

The research problem and the motivation for the problem were provided first. Initial understanding of the problem was obtained by literature studies. Previous results related to the research problem were searched for, studied, and analysed. Resources such as ACM Digital Library, CiteSeer, IEEE Xplore, and SpringerLink were retrieved. Search engines such as Google and Yahoo were used for discovering and crosschecking relevant information. In order to narrow down the research problem, a set of research questions were raised and research hypotheses were formulated to tackle the underlying problem issues.

Step 2: Solution Construction.

Contributions were constructed, which include rules for component construction from legacy systems, concepts, a spectrum modelling language and a framework. To demonstrate the applicability of proposed research, a software environment and related tools were developed and applied in both academic and engineering perspectives.

Step 3: Validation and Verification.

The hypotheses were verified by case studies and validated by means of criteria. Case studies were used to validate and demonstrate the feasibility of the proposed approach, which showed that the methodology can work and produce impressive results. It should be mentioned that it is impossible to judge whether the results are entirely due to the methodology used. A challenge in this respect was to carefully select case studies which are representative and which also span the potential application space of the targeted application domain.

Step 4: Conclusions

Based on the experiences from the evaluation, the applicability of the methods was discussed. As research is an iterative process, new relevant and interesting research questions could be found and formed as the future research topics.

1.3 Research Questions and Hypothesis

1.3.1 Research Questions

This research is driven by a number of problems inherent in the current state of both academic and industrial research area. These problems inform a rationale for the project. The overall research question this thesis tries to answer is:

How to evolve a legacy system into a Pervasive Computing environment?

In order to be able to answer this question, a set of research questions are defined that address the problem in detail.

RQ1: What is the Pervasive Computing?

RQ2: How can the useful service be identified from legacy system?

- What are the rules to extract useful components from legacy system?

- What is the role of WSL in legacy system decomposition and services identification?

RQ3: How can the service integrate in Pervasive Computing environment?

- How to do the pervasive service integration?

RQ4: How can tool support be provided for the proposed approach?

RQ5: Does the legacy system service identification effective and feasible for the legacy software system evolution process in Pervasive computing environment?

RQ6: Does this approach improve the efficiency of Pervasive computing development industry?

1.3.2 Research Hypothesis

The main hypothesis underlying the present thesis is that to evolve a legacy system into a Pervasive Computing environment a cost efficient manner.

H1: None of current techniques alone is sufficient to prevent the legacy crisis, but integration of various techniques can reduce the overall effort required to maintain the ever-increasing amount of legacy software code.

H2: The proposed approach adopts a component-based perspective, assuming that legacy systems are componentised through modular or aspectual decomposition, and supporting gradual migration approach.

H3: Since only parts of the re-engineering process can be automated, a cost efficient and highly industrialised re-engineering approach will have to support the followings:

- Provide a well defined re-engineering methodology.

- Maximise automation, but allow for flexibility and combination with manual approaches.

- Provide a decision framework for automation vs. manual re-engineering, based on cost and resulting quality.

H4: Not all parts of a legacy system are of equal value and only small parts of a system are representing real business rules and process logic.

- This kind of code tends to be very specific to the underlying platform, thus migrating it to a new platform doesn't make a lot of sense.

- Modern platforms usually provide much technical infrastructure functionality, so there is no need for large amounts of custom "glue" code. Meanwhile, a well structured system has to deal with much fewer exceptional cases, again eliminating the need for custom code that deals with them [61].

1.4 Original Contributions

In this thesis, an Agent-based Service-Oriented Approach to Evolving Legacy System into a Pervasive Computing Environment is proposed, which integrates all technical supports into a systematic method for software evolution in Pervasive Computing environment. Concretely, the original contributions of this thesis are as follows:

C1: A unified approach integrates the software evolution approach with the Pervasive Computing technique. It utilises reusable legacy resources into Pervasive Computing environment to build pervasive applications across distributed, dynamic environment and service oriented architecture communities. This research develops an effective way to reuse legacy assets in future Pervasive Computing environment.

C2: Identify service from legacy software systems for the use in Pervasive Computing environment with reverse engineering techniques such as program slicing, software clustering and programme transformation technologies.

C3: Using the service-oriented design approach into the Pervasive Computing environment.

C4: An evolvable agent-based service-oriented architecture can use Multi-agent architecture to integrate the service which identified from legacy system into the Pervasive Computing environment.

1.5 Organisation of Thesis

The thesis is organised as follows:

Chapter 1 defines the research objectives, explains the research characteristics, selects the research method, identifies the research questions, illustrates the research hypothesis, highlights original contributions and defines the success criteria.

Chapter 2 gives a general overview of research background and defines basic concepts related to pervasive computing.

Chapter 3 discusses the related work, which includes software re-engineering techniques, software evolution, multi-agent systems, service-oriented systems and some projects related to Pervasive Computing.

Chapter 4 introduces an agent-based service-oriented approach, to evolve a legacy system into Pervasive Computing environment. The architecture and processes of this approach are proposed.

Chapter 5 describes the service identification for Pervasive Computing environment based on software reverse engineering techniques such as program slicing, software clustering and programme transformation.

Chapter 6 describes the agent-based integration method for Pervasive Computing environment.

Chapter 7 describes the experiments performed on an intelligent building control system, which demonstrates that the proposed approach works in practice and is indeed scalable.

Chapter 8 concludes the thesis. The success criteria are revisited and the future work

is discussed.

Appendix A lists the source code of a Building Control System.

Appendix B lists the source code of TFM13 AST in XML.

Appendix C lists all the related publications by the author during the PhD study.

Chapter 2

Background and Basic Concepts

Objectives

- To present an overview of Pervasive Computing.
- To define basic concepts related to Pervasive Computing.

2.1 Pervasive Computing

2.1.1 Overview of Pervasive Computing

The concept of Pervasive Computing emerged in the early 1990s by Mark Weiser. Pervasive computing is the method of enhancing computer use by making many computers available throughout the physical environment [127], but making them effectively invisible to the user [126]. Pervasive computing names the third wave in computing.

Perhaps the biggest surprise in some pervasive computing scenarios is how simple and basic all the component technologies are. The hardware technologies (laptops, handhelds, wireless communication, software-controlled appliances, room cameras, and so on) are all here today. The component software technologies have also been demonstrated: location tracking, face recognition, speech recognition, online calendars, and so on.

But many Pervasive Computing scenarios seem like science fiction rather than reality till today. The answer lies in the fact that the whole is much greater than the sum of its

parts. In other words, the real research is in the *seamless integration of component technologies* into a system. The difficult problems lie in architecture, component synthesis and system-level engineering. The research which this thesis based on focuses on the evolutionary system component integration, which also be the greatest challenge in the exploring of pervasive computing. Jini technology used in component integration maybe one of the approach in the integration methods, it's just an attempt to achieve the pervasive goal in the development of computing. Because many research organisations represent a broad communal effort to make pervasive computing a reality, there are many useful utilities can be used, both in enterprise and in research field. So we trust, with our effort, the scenarios mentions before will be a reality in the future.

We also should fully aware that attaining the vision of pervasive computing would require tremendous creativity and effort by many people, sustained over many years. The early decades of the 21st century will be a period of excitement and ferment, as new hardware technologies converge with research progress on the many fundamental problems. Pervasive computing offers new beginnings for the adventurous and the restless — a rich open space where the rules have yet to be written and the borders yet to be drawn.

Pervasive computing is fundamentally characterised by the connection of things in the world with computation. Pervasive computing just might help to free the minds from unnecessary work, and connect people to the fundamental challenge that humans have always had: to understand the patterns in the universe and ourselves with them. Weiser heralded a future of ubiquitous and invisible computing in a now often referenced article in Scientific American in which he stated: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." When articulated, this was a vision too far ahead of its time – the hardware technology needed to achieve it simply did not exist.

A pervasive computing system should be pervasive, embedded, nomadic, adaptable, powerful, yet efficient, intentional and eternal. It must everywhere, with every portal reaching into the same information base; live in our world, sensing and affecting it;

allow users and computations to move around freely, according to their needs; provide flexibility and spontaneity, in response to changes in user requirements and operating conditions; free itself from constraints imposed by bounded hardware resources, addressing instead system constraints imposed by user demands and available power or communication bandwidth; enable people to name services and software objects by intent, as opposed to by address; never shut down or reboot; components may come and go in response to demand, errors, and upgrades.

After a decade of technology progress (include new technologies -- such as the Global Positioning System (GPS), smart cards, and radio frequency identification (RFID) tags -- and social developments such as the increasingly widespread acceptance of video surveillance in public places.), many critical elements of pervasive computing are now viable commercial products, such as: handheld and wearable computers; wireless LANs; and devices to sense and control appliances.

Pervasive computing is the trend towards increasingly ubiquitous, connected computing devices in the environment, a trend being brought about by a convergence of advanced electronic -- and particularly, wireless -- technologies and the Internet. Pervasive computing devices are not personal computers as we tend to think of them, but very tiny - even invisible - devices, either mobile or embedded in almost any type of object imaginable, including cars, tools, appliances, clothing and various consumer goods -- all communicating through increasingly interconnected networks. According to Dan Russell, director of the User Sciences and Experience Group at IBM's Almaden Research Centre, by 2010 computing will have become so naturalised within the environment that people will not even realise that they are using computers. Russell and other researchers expect that in the future smart devices all around us will maintain current information about their locations, the contexts in which they are being used, and relevant data about the users.

A number of leading technological organisations are exploring pervasive computing. Xerox's Palo Alto Research Centre (PARC), for example, has been working on pervasive computing applications since the 1980s. Although new technologies are emerging, the most crucial objective is not, necessarily, to develop new technologies.

IBM's project Planet Blue, for example, is largely focused on finding ways to integrate existing technologies with a wireless infrastructure. Carnegie Mellon University's Human Computer Interaction Institute (HCII) is working on similar research in their Project Aura, whose stated goal is "to provide each user with an invisible halo of computing and information services that persists regardless of location." The Massachusetts Institute of Technology (MIT) has a project called Oxygen. MIT named their project after that substance because they envision a future of pervasive computing devices as freely available and easily accessible as oxygen is today. There are other projects related to pervasive computing as below: Project Endeavour at University of California, Berkeley, AT&T Research in Cambridge, U.K. and at the IBM TJ Watson Research Centre. Characteristics of Software Evolution

2.1.2 Pervasive Information Technology

A quote from Sun Microsystems summarises the complexity of a pervasive solution: Pervasive Computing aims at the “Convergence of Computers, Communication, Consumer Electronics, Content and Service”. A generic schema which is applied somehow to most solutions can be simplified as a three tier vertical structure:

- Device: The front-end of information technology is the wide range of pervasive devices, designed for creating and accessing information on the fly. These devices are the most visible interfaces to the user and penetrate our business and all day life.

- Workstation: Workstations form an optional middle tier. The traditional Personal Computer offers capabilities for working with complex information and managing local personal devices. Often, this layer is even omitted, since most pervasive appliances are able to access the provider’s networks directly. Devices like set-top-boxes can replace or complement the personal workstation as a gateway between personal devices and public networks.

- Server: Web servers, enterprise servers and mainframes mainly focus on storing and processing large amounts of information using their strong computing power.

Pervasive Computing introduces significant changes on software products.

Behind this hierarchy of computing systems two underlying layers can be identified, which are of increasing importance:

- Standards: There is a broad basis of common standards on which the information technology is based on. Standards ensure interoperability and connectivity of systems as well as information and application exchange. Since standards are an important issue for Pervasive Computing, they will be ubiquitous through the pervasive computing environment.

- Services: Numerous kinds of services complete the Pervasive Computing landscape. They establish the infrastructure for the widespread usage of computing, since information is intrinsically combined with the accompanying services to provide them.

2.1.3 Pervasive Service

A service refers to a software component that performs computation or action on behalf of a system entity. This entity can be the user or another service. Services are usually well-defined in their functionality as well as their inputs and outputs [50].

We identify the five goals of ubiquity, with regards to a service, as Availability, Transparency, Seamlessness, Awareness, and Trustworthiness (ATSAT) as depicted in the figure. These goals may be satisfied to varying degrees based on user needs and operating conditions.

Figure 2-1. Pervasive Service Model

A pervasive service has features as below (As shown in Figure 2-1):

- Availability: Ideally a ubiquitous service should be available independent context. The service should be also available regardless of changes in user status, needs, and preferences.

- Transparency: According to Weiser, a good tool is an invisible tool. Weisers notion of disappearance, where a tool is "literally visible, effectively invisible" means that the tool does not intrude on the user consciousness; the user focuses on the task, not the tool. Ubiquitous computing provides smarter unconscious, so that users do more easily and intuitively without requiring user attention and awareness of the underlying technology. Transparency implies more than just a user-friendly interface; the technology should facilitate the task in a non-intrusive way and in this way "hide" the underlying technology from the user [16].

- Seamlessness: Seamlessness can be defined as the capability of providing an everlasting service session under any connection with any device. The ultimate goal is that the system will recognise the user wherever she logs on, on any system, with any equipment, at any time, with the applications in a given state and have them adapt in the best possible way given these surrounding conditions. Seams occur when the service fails to satisfy the minimum QoS requirements set

by the end-user [62].

- Awareness: Pervasive devices extend the human senses by providing greater awareness of the surrounding environment. By blending into the physical world, a ubiquitous service bridges the gap between the end-user and his surrounding. We advocate the need for mutual awareness between the user (context) and the service (feedback). Abowd and Mynatt [84] put forth the "five W's" of context, providing a good starting point of the different components that should be put together to provide user context. The five Ws are:- Who (the ability of a device to identify not only its owner, but other people and devices in its vicinity within the environment), What (the ability to interpret user activity and behaviour, and using that information to infer what the user wants to do), Where (the ability to interpret the location of the user and use that to tailor functionality), When (the ability to understand the passage of time, use it to understand the activities around and to make inferences), and Why (the ability to understand the reasons behind certain user actions). In addition to the system awareness of its user, a ubiquitous environment provides user awareness of the task (i.e. feedback) in a way that may enhance the user's decisions.

- Trustworthiness: We define trust of an entity in a ubiquitous service environment as the confidence that the entity will behave as expected in a given context. Mutual trust must be established between different entities in a ubiquitous environment in a sense that each entity is assigned a trust value based on its behaviour. An entity can be a device, a service or a user. In the latter case, the trustworthiness of a service or a device has psycho sociological aspects that affect its usability. The model of trust in a ubiquitous context should capture both the needs of the traditional world of computing where trust is based on identity, and of the world of ubiquitous and pervasive computing where trust is based on identity, physical context or a combination of both [13, 34]. In other words, both identity-based and context-based trust relationships should be defined between different entities within a ubiquitous environment.

2.2 Ubiquitous Technologies

Pervasive computing system is an emerging computing model that provides the ability to perform higher throughput computing by taking advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution across a parallel infrastructure.

Other enterprise development technologies such as DCOM, Enterprise Java Beans (EJB), Java 2 Enterprise Edition (J2EE), and CORBA are all systems designed to enable the construction of distributed applications. They provide standard resource interfaces, remote invocation mechanisms, and trading services for discovery and hence make it easy to share resources within a single organisation.

Comparatively, pervasive computing systems use the resources of many separate computers connected by a network (usually the Internet) to solve large-scale computation problems. Grid provide the ability to perform computations on large data sets, by breaking them down into many smaller ones, or provide the ability to perform many more computations at once than would be possible on a single computer, by modelling a parallel division of labour between processes. Today resource allocation in a Grid is done in accordance with service level agreements. As a type of distributed computing, the Grid should be characterised by other related technologies.

The most widely accepted standard distributed object technologies include: Object Management Group's Common Object Request Broker Architecture (CORBA), Sun's Java remote method invocation (RMI), Sun's Enterprise JavaBeans (EJBs) and Microsoft's Distributed Component Object Model (DCOM). Also, some other related technologies such as P2P, JINI and Web services will be reviewed.

2.2.1 CORBA

In computing, Common Object Request Broker Architecture (CORBA) [65] is a standard for software component, created and controlled by the Object Management Group (OMG). It defines APIs, communication protocol, and object/service information

models to enable heterogeneous applications written in various languages running on various platforms to interoperate. CORBA therefore provides platform and location transparency for sharing well-defined objects across a distributed computing platform.

CORBA uses an interface definition language (IDL) to specify the interfaces that objects will present to the world. CORBA then specifies a "mapping" from IDL to a specific implementation language like C++ or Java [26]. This mapping precisely describes how the CORBA data types are to be used in both client and server implementations. Standard mappings exist for Ada, C, C++, Lisp, Smalltalk, Java, and Python [91].

CORBA is an integration technology, but not a programming technology. As such, it is used to connect distributed objects and integrate them with other heterogeneous computing environments.

2.2.2 J2EE

Java 2 Platform Enterprise Edition (J2EE) technology [29] provides a component-based approach to design, development, and deployment of Web enabled enterprise applications. J2EE offers a multi-tier architecture that separates presentation logic, business logic and back-end services or database. J2EE consists of different components to assist the fast development and deployment of each tier, including:

- Java Applications and applets running on client machines to access to Web servers.
- Java Server Page (JSP) [18] and Java Servlet [58] running on Web servers to receive and responses clients' request.
- Enterprise JavaBeans running on the application servers to implement business logic.

2.2.3 EJB

Enterprise JavaBeans (EJB) [83] technology is the server-side component architecture for J2EE. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.

The EJB specification details how an application server provides persistence, transaction processing, concurrency control, events using Java message service, naming and directory services (JNDI) [31], security (JCE and JAAS), deployment of software components in an application server and remote procedure calls using RMI-IIOP or CORBA. It also defines the roles played by the EJB container and the EJBs as well as how to deploy the EJBs in a container.

EJBs are deployed in an EJB container within the application server. And each EJB must provide a Java implementation class and two Java interfaces. Because these are merely Java interfaces and non-concrete classes, the EJB container must generate classes for these interfaces that will act as a proxy in the client. Client code invokes a method on the generated proxies, which in turn places the method arguments into a message and sends the message to the EJB server. The proxies use RMI-IIOP to communicate with the EJB server.

2.2.4 RMI

The Java Remote Method Invocation [37] API, or Java RMI, is a Java application programming interface for performing remote procedure calls for distributed computing. Because it is implemented in Java, it is platform independent. Similar to CORBA, it generates the stub and skeleton classes for the client and server. Java interface is a natural construct for the interface definition.

There are two common implementations of the interface, the initial one to be implemented known as JRMP and a version compatible with CORBA. Usage of the term RMI may solely denote the programming interface or may signify both the API and JRMP, whereas the term RMI-IIOP, read RMI over IIOP, denotes the RMI interface delegating the most of the functionality to the supporting CORBA implementation. The

original RMI API was generalised somewhat to support different implementations [23]. Additionally, work was done to CORBA, adding a pass by value capability, to support the RMI interface. Still, the RMI-IIOP and JRMP implementations are not fully identical in their interfaces.

2.2.5 DCOM

Distributed Component Object Model (DCOM) [104] is a Microsoft proprietary technology for software components distributed across several networked computers to communicate with each other. It extends Microsoft's COM, and provides the communication substrate under Microsoft's COM+ application server infrastructure. It has been deprecated in favour of Microsoft .NET.

In terms of the extensions it added to COM, DCOM had to solve the problems of marshalling (serialising and desterilising the arguments and return values of method calls "over the wire") and distributed garbage collection (ensuring that references held by clients of interfaces are released when, for example, the client process crashed, or the network connection was lost) [43]. DCOM was a major competitor to CORBA. Proponents of both of these technologies indicate that they will become the model for code and service-reuse over the Internet. However, the difficulties involved in getting either of these technologies to work over Internet firewalls, and on unknown and insecure machines, meant that normal HTTP requests in combination with Web browsers won out over both of them. Microsoft, at one point, attempted and failed to head this off by adding an extra http transport to RPC [75] called "ncacn_http" (Network Computing Architecture, Connection-based, over HTTP).

In a world, DCOM is another object technology integration environment supported only on Microsoft Windows platforms. DCOM-based applications can take full advantage of the existing Microsoft services such as security and transactions. Some software vendors, migrate DCOM to other platforms such as UNIX so that, DCOM may eventually become a truly cross-platform technology.

2.2.6 Peer to Peer

A Peer-to-Peer (or P2P) [85] computer network is a network that relies primarily on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively low number of servers. P2P networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. Sharing content files (see file sharing) containing audio, video, data or anything in digital format is very common, and real time data, such as telephony traffic, is also passed using P2P technology [33].

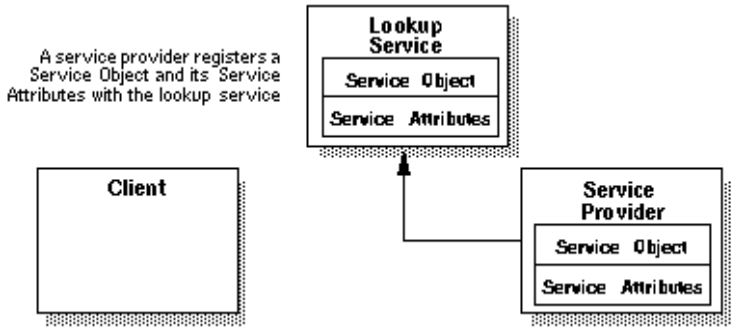
In P2P computing, machines share data and resources, such as spare computing cycles and storage capacity via the Internet or private networks. Machines can also communicate directly and manage computing tasks without using central servers. It is defined as a class of applications that takes advantage of resources storage, cycles, content, human presence available at the edges of the Internet [59].

A pure P2P network does not have the notion of clients or servers, but only equal peer nodes that simultaneously function as both clients and servers to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server. A typical example for a non P2P file transfer is an FTP server where the client and server programs are quite distinct, and the clients initiate the download/uploads and the servers satisfy these requests.

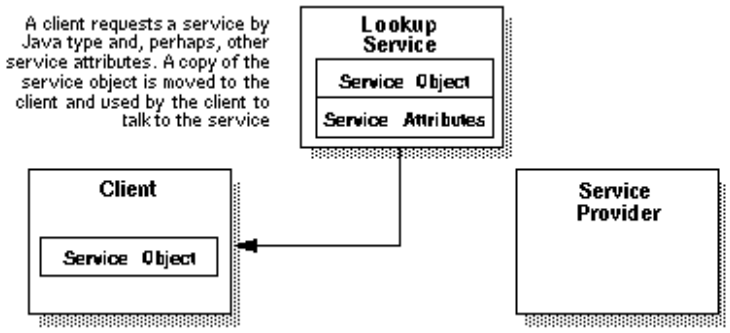
2.2.7 Jini

Jini [68] is a network architecture for the construction of distributed systems where scale, rate of change and complexity of interactions within and between networks are extremely important and cannot be satisfactorily addressed by existing technologies. It is a Java-based infrastructure developed by Sun Microsystems that can provide all the services necessary to support parallel and distributed applications. Jini is primarily concerned with communications between devices. It is designed to provide a software

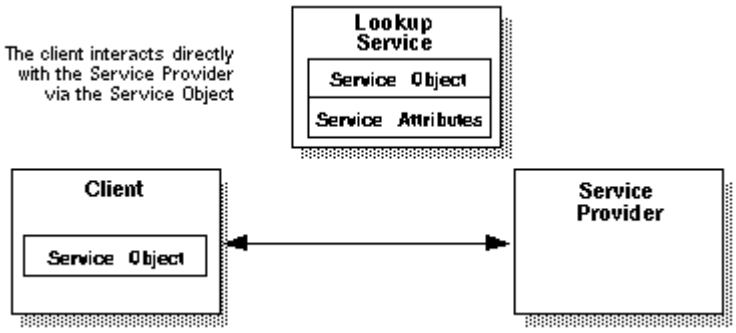
infrastructure that can form a distributed computing environment, which offers network plug and play. Figure 2-2 shows the Jini discovery and lookup protocols.



a. Discovery and Join



b. Lookup



c. Service invocation

Figure 2-2. Jini Discovery and Jini Lookup protocols [2]

Jini technology provides a flexible infrastructure for delivering services in a network and creating spontaneous interactions between clients that use these services regardless of their hardware or software implementations.

Jini network technology is an open architecture that enables developers to create network-centric services (whether implemented in hardware or software) that are highly

adaptive to change. Jini technology can be used to build adaptive networks that are scalable, evolvable and flexible as required in dynamic computing environments.

Jini is a simple set of Java classes (APIs) and services within a distributed computing framework. It allows cooperating devices, services, and applications to access each other seamlessly, to adapt to a dynamic environment, and to share code and configurations transparently.

Jini is a set of specifications that enables services to find each other on a network and allows these services to participate within certain types of operations within the framework. This set of services on a specified network is referred to as the Jini Community. In a Jini Community, everything is a service whether it is hardware or software. It allows these services to interact in a dynamic and robust way: no user intervention when devices are added or removed, adaptability when services come and go, no prior knowledge of the services implementation are needed simplifying administrative duties.

A component integration tool named JAIW is centred by Jini technologies to provide the support to the progress of components integration in the design of evolutionary pervasive computing system.

It provides the following services:

- Automatic resource installation
- Automatic resource discovery
- Intelligent resource management and job scheduling

2.2.8 Web Services

With the rapid global adoption of Business to Business (B2B) [103] and e-Commerce activities worldwide, the concept of the service aims at empowering the business process integration over the Web. To facilitate the use of the Web for business process integration and collaboration between trading partners, the service builds on the top of

components referred to as Web services [30]. Such services offer specific business related functionality, reside in the application servers, can be programmatically integrated with other systems, and perform interactive tasks involving multiple steps on a user's behalf.

To allow for services to interoperate, a common industry standard, such as SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI (Universal Description, Discovery, Integration) has been proposed. SOAP, which is encoded in XML and HTTP, is a message passing mechanism and serves as a uniform invocation mechanism between Web services. The WSDL describes the interface points of the Web services that are further indexed in searchable UDDI.

2.3 Service Oriented Computing

In the past few years, a rapid progress in Service-Oriented Computing (SOC) [57] is witnessed, which represents a paradigm shift from the current mainstream Object-Oriented Computing (OOC) paradigm to the SOC paradigm. This paradigm shift is changing the way that both software and hardware are developed and used. The abstraction and granularity of software parts have been increased from modules, objects, components to services, as reuse and integration technology developed. Service Orientation (SO) is the next step after Object Orientation (OO) and Component-Based Software Development (CBSD) [108]. Service Oriented Computing (SOC) is a software development paradigm in which software systems can be recursively constructed as a set of services that employs standardised service interfaces and interaction protocols.

The desired objectives behind this emerging distributed computing paradigm – service-oriented computing is Software as a Service (SaaS) [78, 130]. Bennett [130] suggests that software will be delivered and consumed as services from a long-term vision for software evolution. Even some software systems are still implemented as tightly coupled systems; they will be invoked via a service-based interface from the usage point of view. For decades, companies developed and maintained their software on their own infrastructure. Software as a Service is a new delivery model where

companies do not own the software but rent the software services. In the SaaS scenario, a software provider is responsible for the availability of software services, which may include maintenance, scalability and disaster recovery. The companies pay a software provider for using the software service.

Service-Oriented Architectures (SOAs), Service Oriented Enterprise (SOE), Service Oriented Infrastructure (SOI), Web Services (WS) and associated protocols and standards have emerged and a solid foundation for the new paradigm is being grounded. Appendix A lists standards related to Web services and service-oriented architectures. Figure 2.3 briefly illustrates the Web service platforms.

Service-Oriented Architectures (SOAs) is the keystone of service-oriented computing. It is really the latest stage in a gradual evolution of ideas that began in the early 1980s within the Object-oriented Programming community and continued into the 1990s in other communities such as DCE, COM/DCOM, CORBA and J2EE. It describes a method for building a corporate software infrastructure that allows different applications to exchange data and processes regardless of the operating systems or programming languages underlying those applications. SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. SOA is particularly applicable when multiple applications running on varied technologies and platforms have to communicate with each other [63]. In this model, an application or a portion of an application is a service that another application or person can subscribe to without extensive custom code. It is an architecture evolution and it affects the software life cycle from the software as a service point of view.

Service-Oriented Architectures (SOAs) are the foundation for application to application communication in distributed computing environment. This architecture promotes interoperability and extensibility among applications, as well as allows them to be combined in order to perform more complex operations. In this architecture, service provider, service requester and service registry are three main roles of Web Services. The service requester is the consumer of a Web Service and is most likely a program running on the server of a business customer. The service requester program gets the information about available services from a UDDI repository, the service

registry. The available Web Services are described in WSDL to support platform-neutral communications. When the service requester has selected a service, it will use the WSDL description to find out how to access the service. Once found, the WSDL description is used to generate a SOAP request message that is sent to the application server, which acts as the service provider.

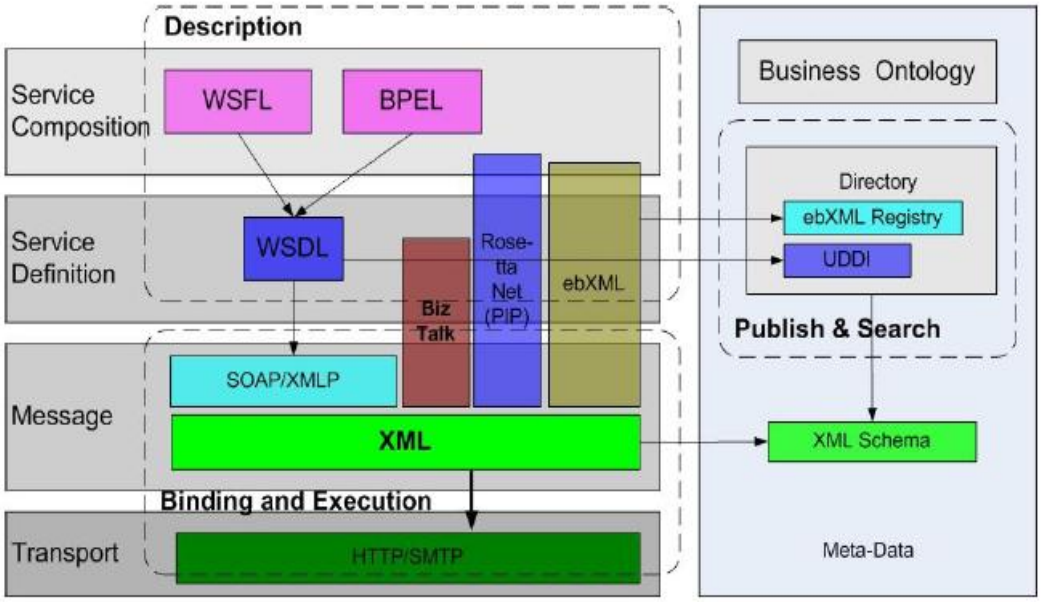


Figure 2-3. Web Service Platform

Based on SOAs, software services comprise service-oriented applications. A software service is an abstract resource that represents a capability of performing tasks that represents a coherent functionality from the point of view of provider entities and requester entities [9].

A service is a coarse-grained, discoverable and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [10]. A collection of services with well-defined interfaces and shared communications model is called a Service-Oriented Architecture (SOA). To be used, a service must be realised by a provider agent. This provider agent is the concrete piece of software (or hardware) that sends and receives messages, while the service is the resource characterised by the abstract set of

functionality that is provided. It processes certain well-defined XML documents what it receives through some combination of transport and application protocols. It is a major construct for publishing and should be used at the point of each significant interface. A software service may be constructed by object-oriented or component-based technology. It may operate as a stand-alone process, or as part of a Web or application server, or as a thin front end for a massive enterprise application. It may be consumed singly or as collaborations.

Software services have various characteristics, such as discoverability and dynamically binding, self-contained, modular and composability. Services stress interoperability with coarse-grained interfaces. Since SOA separates the services implementation from its interface, the service implementation can be adapted easily to accommodate changing needs, which means the system can be evolved without breaking applications that consume the service [136].

SOA is implemented by technologies other than Web services, but the term and concepts have gained popularity recently because of Web services [39]. Web services are software services based on standardised Web interfaces, Web services protocols. Web services exploit XML technology and the Internet to integrate applications that can be published, discovered and used in a technology neutral, standard form over the Web [90]. In fact, Web services are about interoperable document-centric computing [115]. Web services are going to play an important role in the future of distributed computing, significantly leading to a software evolution and a business revolution. It is especially suitable for application integration and legacy system reengineering [99].

Although there are different and often seemingly inconsistent uses of the term Web services, W3C gives a definition that captures the shared essence of the term. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialisation in conjunction with other Web-related standards [9]. Web services based on XML, transported in synchronous or asynchronous forms. They can

be advertised and discovered in a service registry UDDI over Intranet and Internet. The definition of Web Services from IBM is Web services are self-describing, self-contained, modular applications that can be mixed and matched with other web services to create innovative products, processes and value chains. Web services are Internet applications that fulfil a specific task or a set of tasks that work with many other Web services in an interoperable manner to carry out their part of a complex work flow or a business transaction. This definition emphasises the independence of a single service and the interoperability within services. The Web Services Interoperability Organisation (WS-I) seems to have a more generic view: WS-I views Web Services as having general purpose architecture with inherent interoperability that supports B2B scenarios, but extends far beyond in scope and functionality. This definition implicitly carries the notion of support for document-centric, loosely coupled business collaboration. Web services are based on the request-response-model with no business process semantics. All in all, Web services are modular applications that are self-describing and can be published, located and invoked from anywhere on the web or within any local network based on open Internet standards.

Web Services are extremely attractive because they can effectively address the demand for short development cycles, distributed development and global user base [9]. Web Services provide new opportunities for integration of disparate applications and the secure exchange of data and services over the Internet. Web Services define a transport method for processes defined in XML. They are not about providing a user interface to applications and they do not process data. However, at the core of the Web Services evolution is their ability to allow code to speak to code without human intervention. Web Services can provide a standard-based response to data interchange, integration and interoperability challenges. Web Services are based on three main standards. These standards support real-time exchange of data (SOAP), description of Web Services (WSDL) and their publishing to business partners (UDDI). Web Services enable the collaboration and coordination of contractually defined software services distributed over the network and organised in a Service-Oriented Architecture (SOA).

The steps in publishing and using Web Services are as follows. First, a service

provider creates a Web Service. Then the service provider describes the Web Service in a WSDL file. Third, the service provider publishes the Web Services into a service registry based on UDDI. Four, a service requester discovers the Web Service in the registry via UDDI interface. The UDDI registry provides the service requester with a WSDL service description and a URL (Uniform Resource Locator) pointing to the service itself. Finally, the service requester directly binds to the service and invokes it via SOAP according to this information [9]. In a word, Web Services are modular applications that are self-describing and can be published, located and invoked from anywhere on the Web or within any local network based on open Internet standards.

Because of SOA, Web Services and XML share the essential qualities of their long-lived predecessors (details are shown in Figure 2-4), they will not only survive, but also have the potential to change the way that systems are built and connected within and between organisations. As with successful architectures and standards of the past, skilful and appropriate implementation should bring about a phenomenal payoff for an organisation.

Enabled by Web services	
Technology neutral:	Endpoint platform independence
Standardised:	Standards based protocols
Consumable:	Enabling automated discovery and usage
Enabled by SOA	
Reusable:	Use of Service, not reuse by copying of code/implementation
Abstracted:	Service is abstracted from the implementation
Published:	Precise, published specification functionality of service interface, not implementation
Formal:	Formal contract between endpoints places obligations on provider and consumer
Relevant:	Functionality presented at a granularity recognised by the user as a meaningful services

Figure 2-4. Features of Web Services and SOAs

2.4 Summary

In this chapter, the background of Pervasive Computing and some related concepts are introduced.

- A pervasive computing system should be pervasive, embedded, nomadic, adaptable, powerful, yet efficient, intentional and eternal. The generic architecture of pervasive computing based on the evolutionary view of pervasive computing infrastructure.

- Pervasive Computing aims at the "Convergence of Computers, Communication, Consumer Electronics, Content and Service". A generic schema which is applied somehow to most solutions can be simplified as a three tier vertical structure.

- Pervasive service has five goals of ubiquity, with regards to a service, as Availability, Transparency, Seamlessness, Awareness, and Trustworthiness (ATSAT). These goals may be satisfied to varying degrees based on user needs and operating conditions.

Chapter 3

Related Research

Objectives

- To introduce software crisis and legacy system.
- To review the state of the art of software re-engineering approaches.
- To present an overview of software evolution.
- To introduce Wide Spectrum Language and program transformation theory.

3.1 Software Re-engineering and Evolution

No one will doubt today that information systems are business-critical for almost all institutions. However, too much software currently being produced is late, over budget, and does not perform as expected; yet software costs are rising all the time. The fact that the software development industry is in a crisis has been recognised since 1969. Problems associated with the software crisis have been caused by the character of software itself. F. P. Brooks [20] claims the following properties of large software systems:

- Complexity: This is an essential property of all large pieces of software, essential in that it cannot be abstracted away from. This leads to several problems:
  - ✓ Communication difficulties among team members, leading to product flaws, cost overruns, and schedule delays.
  - ✓ It is difficult or impossible to enumerate all the states of the system, which

makes it impossible to understand the system completely.

- ✓ It is difficult to get an overview of the system, so maintaining conceptual integrity becomes increasingly difficult;
- ✓ It is hard to ensure that all loose ends are accounted for.
- ✓ There is a steep learning curve for new personnel.

- Conformity: Many systems are constrained by the need to conform to complex human institutions and systems (e.g., the tax regulations of a state).
- Change: As it is used any successful system will be subject to change to enhance its capabilities, or even apply it beyond the original domain, as well as to enable it to survive beyond the normal life of the machine it runs on and to be ported to other machines and environments.
- Invisibility: For complex software systems there is no geometric representation, as is available to the designers and builders of complex mechanical or electronic machines or large buildings. There are several distinct but interacting graphs of links between parts of the system to be considered (e.g., control flow, data flow, dependency, and time sequence). One way to simplify these, in an attempt to control the complexity, is to cut links until the graphs become hierarchical structures [87].

As one of the most important areas of computer science, software engineering had its origin as a solution to the "software crisis". According to the IEEE Standards, software engineering is defined as [60]:

Software Engineering *is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.*

Software engineering has three elements: (1) methods, which provide the techniques for building software including the design of data structures, program architecture, and

algorithmic procedure, coding, testing, and maintenance; (2) tools, which provide automated or semi-automated support for methods; and (3) processes, the glue that holds the methods and tools together and enables rational and timely development of computer software. A generic view of software engineering can be obtained by examining the process of software development [135].

3.1.1 Legacy Software System

Many methods and techniques have been hailed as the solution to the software crisis; however in practice only small gains in productivity have been achieved and few of these methods pay any attention to the problems of maintaining and enhancing the developed software. Estimates show that 65-75% of total software costs are subsumed in maintenance activities [106]. This number has undoubtedly increased and is increasing at an accelerating rate. The result is that even if the promised large improvements in development speed by the use of new methods do eventually appear they will have little impact on total software costs since any gain from increased development will be swallowed up by the increased maintenance cost [117]. Concern is growing that the development of new software is outpacing the ability to maintain it. In the current decade, four out of seven programmers are working on enhancement and repair projects. With large portions of software budgets being devoted to maintenance, few resources remain for new development. If these trends continue, eventually no resources will be left to develop new systems, and people will enter the Middle Ages of the information age, referred as “legacy crisis” [103].

The term “legacy system” describes an old system which remains in operation within an organisation [123]. Organisations are in fear of their legacy systems. They are afraid of keeping them, since maintaining them is a significant drain on the organisation's resources. They are also afraid of replacing them. A major reason is that those legacy systems are enormously valuable assets. Having stood the test of time and evolved, they provide the most accurate statement of current business practices. Legacy systems, then, are not an issue that will simply go away. Because legacy software systems are so critical to an organisation’s survival, they are a very real problem facing many

organisations and, thus, people need to develop an appropriate strategy for dealing with them. Formal methods can be defined as mathematically based languages, techniques, and tools for specifying and verifying systems. Baumann [17] states that reverse engineering methods must be based on a sound foundation, which entails formal denotation semantics, because if these methods should extract the wrong information during reverse engineering process, this wrong information could lead to new errors in the re-engineered programs. Formal methods can also increase the understanding of a system by revealing inconsistencies, ambiguities, and incompleteness that might go undetected [25]. In the area of reverse engineering, formal methods have been put forward as a means to

- formally specify and verify existing systems in particular those already operating in safety-critical applications,
- introduce new functionalities, and/or
- take advantage of the improvement in systems design techniques [74].

Formal methods can be classified into the following five classes or types, i.e., Model-based, Logic-based, Algebraic, Process Algebra and Net-based (Graphical) methods, which should consist of some essential components: a semantic model, a specification language (notation), a verification system/refinement calculus, development guidelines and supporting tools [135]:

- The semantic model is a sound mathematical/logical structure within which all terms, formulas and rules used have a precise meaning. The semantic model should reflect the underlying computational model of the intended application.
- The specification language is a set of notations which are used to describe the intended behaviour of the system. This language must have a proper semantics within the semantic model.
- Verification system/refinement calculi are sound rules that allow the verification of properties and/or the refinement of specifications.

- Development Guidelines are steps showing the use of the method.
- Supporting tools involve proof assistant, syntax and type checker, animator, and prototype.

There are at least two advantages of using formal methods as the foundation of software reengineering. First, formal methods can help software engineers to acquire a rigorous and precise description of the system being reengineered, therefore greatly increasing the quality of the new system. Second, automation is one of the key goals of reengineering. By applying formal methods, it is possible to automate more of the process of reengineering [135].

3.1.2 Software Re-engineering

There is no commonly accepted definition for the term software reengineering and the related terminology is not standardised. However, there exist various valid definitions of software reengineering that represent different point of views and perspectives. For instance, Chikofsky and Cross define it as ―the examination and alteration of a subject system to reconstitute it in a new form and subsequent implementation of that form in their landmark paper [28]. Afterwards, Arnold defines software re-engineering in his one-volume guide to the reengineering literature [92] as software reengineering is any activity that 1) improves one‘s understanding of software, or 2) prepares or improves the software itself, usually for increased maintainability, reusability or evolvability.

In this definition, the term - software, in addition to source code, includes documentation, graphical pictures and analyses. The analyses are about source code, design, specifications, test data and other documents directly supporting software development or maintenance. Part 1 of this definition includes activities such as browsing, measuring, drawing pictures of software, documenting and analysing. Part 2 includes activities designed to improve static qualities of software, usually so the software is easier for people to work with. Reverse engineering pertains to part 1 of the reengineering definition. Reverse engineering generates information about a software representation (such as source code) to help one understand it or to facilitate its

processing. Furthermore, reengineering is also the general term for activities during corrective, adaptive, perfective or preventive software maintenance.

Software reengineering is significant in system evolution. System replacement is expensive, but system reengineering is often cheaper. Furthermore, system reengineering reduces the risk of losing any critical information, which is embedded in legacy assets. In a word, system reengineering provides lower costs, reduced risk, better use of existing staff, revelation of business rules and incremental development. Software reengineering activities may include views of software, information base, decomposition, composition and transformation. A software view is a representation of software or a report about software. A software view may be for human viewing or not, but it typically is a significant interim representation of software that humans may want to see. Software reengineering and related technology may be viewed as transforming information in one software view to information in another software view. Transformation is central to reengineering and related terms. The transformation may move information into and out of the information base. Key elements of transformations are views of software, an information base, decomposition of software information into objects and relationships in an information base and composition of views from information in the information base.

The software reengineering technology includes many techniques to enhance software, to understand software and to manage knowledge about software. For software improvement, reengineering technologies include restructuring, re-documenting, annotating, updating documentation, reuse engineering, re-modularisation, data reengineering, business process reengineering, maintainability analysis, portfolio analysis and economic analysis. For software comprehension, reengineering technologies include browsing, analysing, measuring, reverse engineering, design recovery. For managing knowledge about software, reengineering technologies include decomposition, reverse engineering, design recovery, object recovery, program understanding, knowledge bases and transformations.

The software reengineering process takes many forms, depending on its objectives. Sample objectives are code refactoring, re-documentation and platform migration.

Reengineering software for reuse in service-oriented computing environment is an important one in these objectives [114].

The problem of exposing existing software systems as services in a service-oriented architecture can be considered as a specific task of legacy system modernisation. Modernising a legacy system is an option whenever the system requires more pervasive changes than those possible during ordinary maintenance, but it still has business value that must be preserved [89]. As to the technical problem, methods, techniques and tools that allow the service-oriented migration process to be carried out effectively, Zhang and Yang [136] have identified three classes of approaches for integrating legacy systems in SOAs: the first class comprises black-box reengineering techniques, which integrate system via adaptors that wrap legacy code and data and allow the application to be invoked as a service. A second class includes white-box methods, which require code analysis and modification for obtaining the code components of the system to expose as Web services, while a third class of grey-box techniques combine wrapping and white-box approaches for integrating parts of the system with valuable business value.

From previous experience, it may depend on the features and the existing environment of legacy systems to apply these modernisation techniques, but it is believed that the software reengineering technique is the core technique for legacy system modernisation in the service-oriented computing environment [108]. In addition, this software reengineering technique combines program comprehension and wrapping techniques, which is regard as a grey-box modernisation.

Software reengineering technology is a practical solution for the problem of evolving existing computing systems. Due to the rapid development of computer hardware and software, the demands and costs of software changes are increasing continuously. The phases of software reengineering include:

- Reverse engineering is the process of analysing a subject system to (1) identify the system's components and their interrelationships and (2) create representations of the system in another form or higher level of abstraction [131]. The reverse

engineering includes the following research areas:

- Reengineering requirements involves the planning phase where the reengineering objectives are clearly identified within the context of the selected legacy system that needs be migrated [129].

- Analysis involves a wide variety of source code analysis and abstraction models to assess the technical, functional and architectural aspects of the existing systems [15].

- Positioning focuses on restructuring a system to enhance its qualities, or making the program more readable without changing its external behaviour.

- Re-documentation is a form of restructuring a system with a semantically-equivalent representation in a different view in order to facilitate understanding.

- Design recovery [96] aims to identify meaningful higher-level abstractions of a system and associates code segments with specific functionality.

➢ Forward engineering is the traditional process of moving from high-level abstractions and logical, implementation-independent designs to the physical implementation of a system. It aims to improve a software system by considering new functional and non-functional requirements for the migrant system. Forward engineering follows a sequence of activities, including elicitation of new requirements, transformation of the system, and finally deployment of the system in its new environment.

3.1.3 Software Evolution

Software evolution is the process of adapting an existing software system to conform to an enhanced set of requirements. Software reengineering [51] is software evolution performed in a systematic way [77]. In particular, Chikofsky and Cross define reengineering to be “the examination and alteration of a subject system to reconstitute it

in a new form and the subsequent implementation of the new form". Altering existing systems comprises the majority of all software development time [47] and expense, and evolution comprises the majority of system alteration (maintenance) activities [100].

The major difference between initial development and evolution has to take into account the existing version of the system being evolved [95]. The important concerns including making sure that the new requirements are consistent with those of the existing version, trying to maintain control of the architecture of the system, understanding the code of the current version, and suggesting how the enhancement might be made while maintaining the conceptual integrity of the design. It will help the legacy system to migrate to a new language, platform, operating system, hardware; migrate to a new software development paradigm; increase maintainability and integrate with other systems [27].

Software evolution is the process of conducting continuous software reengineering. Reengineering implies a single change cycle, but evolution can go on forever. In other words, for a large extent, software evolution is repeated software reengineering.

3.1.4 Component-based Software Engineering

Compared to formal methods, cognitive methods rely mainly on domain knowledge. In order to jump from one level up to another abstract level in the process of reverse engineering. One has to throw away some information. No method can guarantee that such a throwing away of information is appropriate [86]. This implies that the abstraction is creative work. In order to achieve correct and practical abstraction, a knowledge base is necessary.

A cognitive model describes the mental process or faculty of knowing a software system [12]. A hierarchy of cognitive design elements to support the construction of a mental model was defined in [7], which explains how to improve program understanding by supporting the actions of identifying software artefacts and the relations between them, by browsing code in delocalised plans, and by building abstractions. These actions comprise canonical reverse-engineering activities.

Two common approaches to program understanding are a functional approach emphasising cognition by what a system does and a behavioural approach emphasising how a system performs [8].

- The functional approach is bottom up and deductive, relying more on the knowledge of the implementation domain to produce higher level of abstractions that may map to the application domain and the system's functional requirements.

- The behavioural approach is top down and inductive, using hypothesis postulation and refinement to match artefacts derived from knowledge of the application domain onto the related software system.

3.1.5 Software Re-engineering Technique

Compared to formal methods, cognitive methods rely mainly on domain knowledge. In order to jump from one level up to another abstract level in the process of reverse engineering. One has to throw away some information. No method can guarantee that such a throwing away of information is appropriate [66]. This implies that the abstraction is creative work. In order to achieve correct and practical abstraction, a knowledge base is necessary.

A cognitive model describes the mental process or faculty of knowing a software system [98]. A hierarchy of cognitive design elements to support the construction of a mental model was defined in [132], which explains how to improve program understanding by supporting the actions of identifying software artifacts and the relations between them, by browsing code in delocalised plans, and by building abstractions. These actions comprise canonical reverse-engineering activities.

Two common approaches to program understanding are a functional approach emphasising cognition by what a system does and a behavioural approach emphasising how a system performs [67].

- The functional approach is bottom up and deductive, relying more on the knowledge of the implementation domain to produce higher level of abstractions

that may map to the application domain and the system's functional requirements.

- The behavioural approach is top down and inductive, using hypothesis postulation and refinement to match artifacts derived from knowledge of the application domain onto the related software system.

3.1.5.1 Software Clustering

Clustering [94] is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics. Clustering is the classification of similar objects into different groups [70], or more precisely, the partitioning of a data set into subsets (clusters) [55], so that the data in each subset share some common trait (often proximity according to some defined distance measure) [11].

Clustering algorithms can be hierarchical or partitional. Hierarchical algorithms [133] find successive clusters using previously established clusters, whereas partitional algorithms determine all clusters at once [111]. Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down) [116]. Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

Clustering analysis is to group large mounts of entities in a dataset into clusters according to their relationship and similarity [41]. Software clustering technique is applied to capture reusable legacy code segments [38], which is independent and loose coupling [128]. It is also applied in program understanding area, such as re-modularisation [102] and component recovery [54]. It is very useful to renovate legacy systems. Clustering methods can be used to identify objects in procedure based legacy systems [56].

3.1.5.2 Program Slicing

Mostly, legacy systems are huge and complex. Slicing a system into many small parts may receive many benefits, including: more flexibility and more reliability, as well as more reusability. Program slicing is a software maintenance technique used to identify all program code that can in any way affect the value of a given variable. The following paragraph informally describes this computation. It is an established technique for reverse engineering and other analyses like testing or debugging. It is available in research prototypes and even in commercial products [46].

Program slicing [110] was defined by Weiser in 1979 as an presented an approach to compute slice by computing consecutive sets of indirectly relevant statement according to data flow and control flow dependences [42]. Only statically available information is used for computing slices, this type of slice is referred to as static slice. Dynamic program slicing [14] can be produced for a given input by maintaining a runtime representation of the syntax tree and marking nodes as their corresponding constructs are executed [76].

There are two main approaches of slicing: The original slicing technique from Weiser [69] is based on traditional data flow analysis [124]; the other approach is based on program dependence graphs (PDG) [64]. Extensive evaluations of different slicing algorithms have not really been done yet for control flow graph based algorithms that some data reported by Atkinson and Griswold can be found in [40]. The only evaluation of program dependence based algorithms that the author is aware of has been conducted by Agrawal and Guo, who just compare two algorithms [97]. Slicing identifies statements in a program which may influence a given statement (the slicing criterion) [32], but it cannot answer the question why a specific statement is part of a slice [71].

3.1.5.3 Wrapping Technique

Wrapping is a practice that transforms a component’s software interface from one form to another. These techniques aim on enhancing interoperability and system flexibility through metadata [137].

The process of wrapping involves different techniques depending on the accessible elements of a legacy system. In the best case scenario, accessing to legacy system source code is available. In this case, it is possible to integrate the legacy system directly to the object wrapper code.

Wrapping technology is used to remove mismatches between the interface exported by a software artefact and the interfaces required by current integration practices. Wrapping technology can modernise legacy system at user interface, data, or the functional (logic) level [52]. The user interface (UI) is the most visible part of a system. UI wrapping improves usability and is greatly appreciated by final users. Data wrapping enables accessing legacy data using a different interface or protocol than those for which the data was designed initially. Data wrapping improves connectivity and allows the integration of legacy data into modern infrastructures. In contrast with data wrapping, functional wrapping not only encapsulates the legacy data, but also the business logic embedded in the legacy system.

UI wrapping consists of wrapping old, text based interfaces with new graphical interfaces. This technique can be extended easily, enabling one new UI to wrap a number of legacy systems. Screen scraping is effective for stable systems where the principle objective is to improve usability. However, the new system is inflexible and difficult to maintain as the legacy system.

Alternative, a XML server has been designed to bridge legacy system to XML in the XML-based B2B architecture. The XML server acts as the contact point between the corporate infrastructure and the rest of the world. The XML server communicates by various means with the internal infrastructures including ERP systems, databases, EDIs, etc. On the other hand, the server interoperates with external organisation by exchanging XML messages.

Data Wrapping: Database Gateway is a specific type of software gateway that translates between two or more data access protocols. It normally translates a vendor-specific access protocol into one of the standard protocols, such as Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC). Using a

database gateway to access legacy data improves connectivity, enables remote access, and supports the integration of legacy data with modern systems. Integration is another data wrapping technology. It uses XML server [58] to communicate with the internal infrastructures and external organisation by exchanging XML messages. It acts as the contact point between the corporate infrastructure and the rest of the world. In addition, most of the commercial XML servers support a wealth of communication protocols and this enables cost-effective integration with the most usual legacy applications. Database replication is also used for data wrapping. It is the process of copying and maintaining database objects in multiple databases that make up a distributed database system. Database replication is often used to enable decentralised access to legacy data stored in mainframes. New applications using the data receive the benefits of local access to a modern database instead of the problems of remote access to an obsolete data repository.

CGI Integration: The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP (HyperText Transfer Protocol) or Web servers. Legacy integration using the CGI is often used to provide fast Web access to existing assets including mainframes and transaction monitors. CGI integration not only wraps the old user interface, but also communicates with the core business logic or data of the legacy system. It is more flexible than screen scraping because the new interface does not need to match the old user interface. However, it still does not fully address maintenance issues.

Object-Oriented Wrapping: The conceptual model of object-oriented wrapping is deceptively simple: individual applications are represented as objects; common services are represented as objects; and business data is represented as objects. In reality, object-oriented wrapping is far from simple and involves several tasks including code analysis, decomposition, and abstraction of the Object-Oriented (OO) model. Among the multiple technical difficulties involved in wrapping a legacy system, two have special relevance: the definition of appropriate object-level interfaces and the need for integrated infrastructure services.

Component Wrapping: Component wrapping is very similar to OO wrapping, but

components, in contrast with objects, must conform to a component model. This constraint enables the component framework to provide the component with quality services [24].

In [105], wrapping is classified into five different levels, namely, job level, transaction level, program level, module level and procedure level. At process level and transaction level, wrappers encapsulate a batch of legacy executive processes. Legacy applications are invoked through wrappers by creating the requested new process and directly deploying the corresponding service without prior knowledge of the corresponding legacy code. In the application level, the wrapper encapsulates only one process. By contrast, at the module level and procedure level encapsulation focuses on clear interfaces, restructured and re-modularised legacy code.

3.2 Program Transformation and WSL

Wide Spectrum Language (WSL) [72, 118, 120-122, 135] has been developed for a number of years and has been used to build a general approach and a tool for addressing research issues such as program comprehension and reverse engineering using program transformation and abstraction techniques. WSL is built on formal methods and supports both object oriented and structural elements of software systems. In order to be successful in reverse engineering, WSL meets at least three conditions:

- WSL must be applicable to parts of the programming language that is relevant for reverse engineering methods.
- WSL should support standard approaches to program analysis such as control-flow analysis, data-flow analysis, etc.
- WSL should be easily and efficiently implementable.

WSL was developed with several advantages in mind:

- The ability to express general specifications in terms of mathematical logic with suitable notation.

- A well-developed library of proven transformations that do not require the user to fulfil complex proof obligations before these transformations can be applied.
- Techniques to bridge the “abstraction gap” between specifications and programs.
- The ability to scale to large programs and applicability to real programs.

The WSL language is built up in a series of stages or levels, starting with a very small and mathematically tractable kernel language. The "Spectrum" in "Re-engineering Wide Spectrum Language" refers to the range of operations, from "low level" things, such as program structures and commands, to high level operations, such as specification statements. By translating a legacy system’s source code to WSL as an intermediate representation, it allows the re-engineering effort to be divided up into smaller steps rather than as a monolithic source to target domain re-engineering effort [82].

3.2.1 WSL Kernel Language

The WSL kernel language is based on infinitary first order logic, which is an extension of ordinary first order logic which allows conjunction and disjunction over (countably) infinite lists of formulae, and quantification over finite lists of variables.

Expressions and conditions (formulae) in WSL are taken directly from infinitary first order logic. Statements in the kernel language are constructed by combining infinitary logic formulae, lists of variables and statement variables. Four primitive statements and three compound statements are needed to define the whole kernel language. Let *P* and *Q* be any infinitary logical formulae and *x* and *y* be any finite, non-empty lists of variables. The primitive statements are [122, 135]:

- **Assertion**: *N* is an assertion statement which acts as a partial skip statement. If the formula *P* is true then the statement terminates immediately without changing any variables, otherwise it aborts (Abnormal termination and non-termination are treated as equivalent, so a program which aborts is equivalent to one which never terminates);

- **Guard**: *[Q]* is a guard statement. It always terminates, and enforces *Q* to be true at this point in the program without changing the values of any variables. It has the effect of restricting previous nondeterminism to those cases which will cause *Q* to be true at this point. If this cannot be ensured then the set of possible final states is empty, and therefore all the final states will satisfy any desired condition (including *Q*);

- **Add variables**: *add(x)* first ensures that the variables in *x* are in the state space (by adding them if necessary) and then assigns arbitrary values to the variables in *x*. The arbitrary values may be restricted to particular values by a subsequent guard;

- **Remove variable**s: *remove(y)* ensures that the variables in *y* are not present in the state space (by removing them if necessary).

The compound statements are:

- **Sequence**: *(S1; S2)* executes *S1* followed by *S2*;

- **Nondeterministic choice**: *(S1 Π S2)* chooses one of *S1* or *S2* for execution, the choice being made nondeterministically;

- **Recursion**: *(μX.S1)* where *X* is a statement variable (a symbol taken from a suitable set of symbols). The statement *S1* may contain occurrences of *X* as one or more of its component statements. These represent recursive calls to the procedure whose body is *S1*.

3.2.2 Extensions to Kernel Language

The kernel language is particularly elegant and tractable but is too primitive to form a useful WSL for the transformational development of programs. For this purpose it is needed to extend the language by defining new constructs in terms of the existing ones using definitional transformations, which consists of the following constructs [135]: Sequential composition; Deterministic choice; Specification statement; Simple

assignment; Nondeterministic choice; Deterministic iteration; Nondeterministic iteration; Initialised local variables; Counted iteration and Block with procedure calls.

A series of new language levels is built up, with the language at each level being defined in terms of the previous level. Each new language level automatically inherits the transformations proved at the previous level, which form the basis of a new transformation catalogue. This technique has proved extremely powerful and has led to the development of a practical transformation system that implements a large number of program transformations.

3.2.3 Extensions to Object Orientation

In order to support object oriented technology, the WSL was extended to include such constructs as class structure which contains both variables (attributes) and procedures (methods or operations) [74, 79].

The syntax of extended WSL adds the following object-oriented portion:

1. Class Definition

```
Class T
  Var
    Ti : xi;    /* Attributes of Class */
  Proc
    mj(In pinjk:Tk, Out poutjl:Tl)    /* Methods of Class */
      Begin
        Aj; /* WSL statements */
      End
  End
```

This statement is the class building declaration. It defines a class named T, which has data fields x_i of type T_i and methods m_j. pin_{jk} stands for the input parameters of method m_j, and $pout_{jl}$ stands for the output parameters of method m_j.

2. Class Hierarchy

```
T Extends T'
```

This statement is used to build the object hierarchy. It declares that class T is a subclass of class T'. Therefore, T inherits the properties of T'.

3. Field Reference

x.d

This is object field reference. x is an object and d is a field of x.

4. Method Invocation

x.m (**In** ek, **Out** yl)

This invokes the method m in object x.

5. Object Declaration

T : x

This statement defines x as a variable of type T. If T is a class, x will be an object of class T.

3.2.4 WSL Related Tools

3.2.4.1 Maintainer's Assistant

One of the most important successes of Maintainer's Assistant (MA) [21, 134, 135] is that it is based on a wide spectrum language, which defines syntax and semantics formally. Maintainer's Assistant (MA) employs transformation techniques to derive a specification from a section of code and to transform a section of code into a logically equivalent form. MA has features as follows:

- It acts, initially, on existing program code as a tool to aid comprehension (possibly by producing specifications) and only the program code is required for the processing;
- The system can work with any language by first translating, i.e., with a standalone translator into WSL and changes are made to the WSL program by means of transformation;
- The system incorporates a large, flexible catalogue of transformations. The applicability of each transformation is tested before it can be applied;

- The system is interactive and incorporates an X-Windows front end and pretty printer called the Browser;
- The system includes a database structure to store information about the program being transformed, such as the variables assigned to within a given piece of code;
- The system includes a facility to calculate metrics for the code being transformed.

3.2.4.2 FermaT

Maintainer's Assistant has evolved into an industrial-strength re-engineering tool, FermaT [118, 119, 135], which allows transformations and code simplification to be carried out automatically. The FermaT tool was also designed to use WSL and has applications in the following areas:

- Improving the maintainability of existing mission-critical software.
- Translating programs into modern programming languages. FermaT often translates program written in obsolete assembler language to more modern languages such as C.
- Extracting reusable components from the current system, deriving their specifications, and storing the specifications, implementation, and development strategy.
- Reverse engineering existing systems to high-level specifications, followed by subsequent re-engineering and evolutionary development.

3.2.4.3 TAGDUR

TAGDUR (Transformation and Automatic Generation of Documentation in UML through Re-engineering) [79-81], was designed to overcome the lack of documentation problem often faced by legacy systems whose original documentation has been lost. By utilising information acquired during the transformational process and by parsing the

code of the transformed system, this tool generates UML diagrams of the transformed system.

3.3 Ongoing Pervasive Computing Projects

A number of leading technological organisations are exploring pervasive computing. Xerox's Palo Alto Research Centre (PARC), for example, has been working on pervasive computing applications since the 1980s. Although new technologies are emerging, the most crucial objective is not, necessarily, to develop new technologies. IBM's project Planet Blue, for example, is largely focused on finding ways to integrate existing technologies with a wireless infrastructure. Carnegie Mellon University's Human Computer Interaction Institute (HCII) is working on similar research in their Project Aura, whose stated goal is "to provide each user with an invisible halo of computing and information services that persists regardless of location." The Massachusetts Institute of Technology (MIT) has a project called Oxygen. MIT named their project after that substance because they envision a future of pervasive computing devices as freely available and easily accessible as oxygen is today. There are other projects related to pervasive computing as below: Project Endeavour at University of California, Berkeley, AT&T Research in Cambridge, U.K. and at the IBM TJ Watson Research Centre. Characteristics of Software Evolution

3.3.1 Oxygen (MIT)

The core of Oxygen is “human-centred”. For over forty years, computation has centred about machines, not people. In the future, computation will be human-centred It will be freely available everywhere, like batteries and power sockets, or oxygen in the air people breathe [3].

New systems will boost the productivity. They will help people automate repetitive human tasks, control a wealth of physical devices in the environment of people living, find the information people need (when people need it, without forcing eyes to examine thousands of search-engine hits), and enable people to work together with other people

through space and time. To support highly dynamic and varied human activities, the Oxygen system must master a number of technical challenges. It must be accessible anywhere. It must adapt to change, both in user requirements and in operating conditions. It must never shut down or reboot — components may come and go in response to demand, errors, and upgrades, but Oxygen as a whole must be available all the time.

The approach of Oxygen is integrated technologies that address human needs. Oxygen enables pervasive, human-centred computing through a combination of specific user and system technologies. Oxygen's user technologies directly address human needs. Speech and vision technologies enable people to communicate with Oxygen as if they are interacting with another person, saving much time and effort. Automation, individualised knowledge access, and collaboration technologies help people perform a wide variety of tasks that people want to do in the ways they like to do them. Oxygen's system technologies dramatically extend the range by delivering user technologies to people at home, at work, or on the go. Computational devices, called Enviro21s (E21s), embedded in our homes, offices, and cars sense and affect the immediate environment. Hand-held devices, called Handy21s (H21s), empower people to communicate and compute no matter where they are. Dynamic networks (N21s) help the machines locate each other as well as the people, services, and resources people want to reach.

The Oxygen technologies work together and pay attention to several important themes:

- Distribution and mobility — for people, resources, and services.
- Semantic content — what we mean, not just what we say.
- Adaptation and change — essential features of an increasingly dynamic world.
- Information personalities — the privacy, security, and form of our individual interactions with Oxygen.

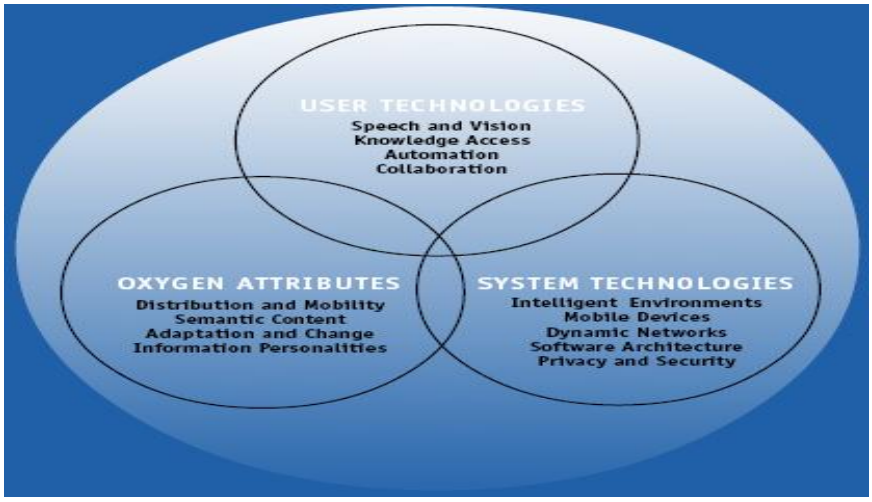


Figure 3-1. Oxygen Integrated Technologies [3]

Oxygen is an integrated software system that will reside in the public domain. Its development is sponsored by DARPA and the Oxygen Alliance of industrial partners, who share its goal of pervasive, human-centred computing. Realising that goal will require a great deal of creativity and innovation, which will come from researchers, students, and others who use Oxygen technologies for their daily work during the course of the project. The lessons they derive from this experience will enable Oxygen to better serve human needs.

Oxygen technologies are entering the everyday lives. Here are some of the technologies being tested at MIT and by the Oxygen industry partners.

Distribution and Mobility: The Cricket location support system provides an indoor analogue of GPS. The Intentional Naming System (INS) provides resource discovery based on what services do, rather than where they are located. The Self-Certifying (SFS) and Cooperative (CFS) File Systems provide secure access to data over untrusted networks without requiring centralised control.

Perceptual interfaces. Multimodal systems enhance recognition of both speech and vision. Multilingual systems support dialogs among participants speaking different languages. The SpeechBuilder utility supports development of spoken interfaces. Person tracking, face, gaze, and gesture recognition utilities support development of visual

interfaces. Systems that understand sketching on white boards provide more natural interfaces to traditional software packages.

Semantic content: Haystack and the Semantic Web support personalised information management and collaboration through metadata management and manipulation. ASSIST helps extract design rationales from simple sketches.

Security and privacy: Trusted software proxies provide secure, private, and efficient access to networked and mobile devices and people. Decentralisation in Oxygen aids privacy: users can locate what they need without having to reveal their own location.

Software and hardware architectures: MetaGlue is a robust architecture for software agents. The GOALS ystem integrates software services to accomplish user-defined goals. RAW and Scale expose hardware to compilers, which optimise the use of circuitry and power. StreamIt provides a language and optimising compiler for streaming applications.

3.3.2 Aura (Carnegie Mellon University)

The most precious resource in a computer system is no longer its processor, memory, disk or network. Rather, it is a resource not subject to Moore's law: User Attention. Today's systems distract a user in many explicit and implicit ways, thereby reducing his effectiveness.

Project Aura will fundamentally rethink system design to address this problem. Aura's goal is to provide each user with an invisible halo of computing and information services that persists regardless of location. Meeting this goal will require effort at every level: from the hardware and network layers, through the operating system and middleware, to the user interface and applications.

Project Aura will design, implement, deploy, and evaluate a large-scale system demonstrating the concept of a “personal information aura” that spans wearable, handheld, desktop and infrastructure computers.

Project Aura’ goals are:

- reduce user distraction
- trade-off plentiful resources of Moore's law for human attention
- achieve this scalably for mobile users in a failure-prone, variable-resource environment

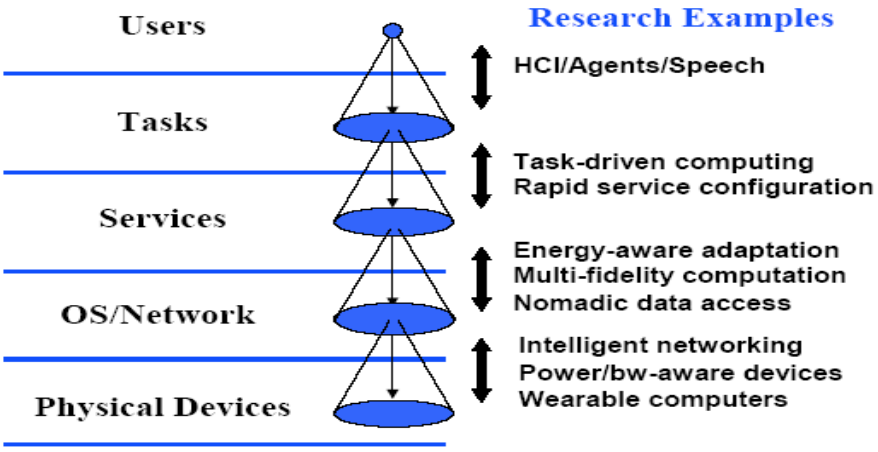


Figure 3-2. Project Aura Research Framework [6]

3.3.3 Endeavour (University of California, Berkeley)

The Project Endeavour, named for the ship Captain Cook sailed on his explorations of the Pacific. The project envision "fluid" information systems that are everywhere and always there, with components that "flow" through the infrastructure, "shape" themselves to adapt to their usage, and cooperate on the task at hand. The researchers in Project Endeavour seek to develop a pervasive Information Utility, based on a new technology of fluid systems, enabling new approaches for problem solving and learning.

The view of the future demands a quantum change in information technology research: dynamic adaptation, self-organisation, and personalisation on a truly massive scale. Its scale and grandeur, its rapid evolution and the radical modes of its use, the heterogeneous nature of its component subsystems and their contained information, and the broad activities it supports, make our envisioned Information Utility enormously complex. This can only be managed by the system implicitly organising its contents

based on tacit in-formation extracted from the environment. Enabled by ubiquitous communication and comprehensive interoperation, implicit organisation makes the utility more human-centred, by raising the level at which users interact with information.

- Personal Information Management is the Killer Application: Computer usage is shifting from corporate processing to the management, analysis, aggregation, dissemination, and filtering of information for the individual in all aspects of their lives.

- People Create Knowledge, Not Data: Information technology has traditionally focused on managing and retrieving explicitly entered data. We will soon be digitising and archiving the bulk of human activity. It will be an on-going stream culled from everyday human activities and physical phenomena. The analysis, processing, and organisation of this information must become more automated.

- Information Technology is a Utility: Access to information cannot stop because a computer has crashed, a link is stale, or a subset of information in not entirely consistent. The challenge is to develop an architecture that continuously provides service on top of highly dynamic underlying information. Flexible architectures are loosely organised and adaptive, allowing great confidence in its operation, and providing administrative scalability (i.e., the administration of a complex system on a planetary-scale by multiple, unaffiliated people with unique objectives, while its physical and logical resources are constantly changing).

- Beyond the Desktop: When millions of people are conducting their efforts on-line with shared information, the confluence of their actions becomes a powerful means of enhancing our productivity and extracting knowledge from information. The challenge is to automatically infer relationships among information, delegate control, and establish authority from available information, assisted with new ways to interact with information and people to enhance human productivity.

The first voyage will develop the initial conceptual architecture and proof of concepts in four critical areas: Information Devices, Information Utilities, Applications, and Design Methodology. This first expedition is organised into a base program and eight optional tasks. The former focuses on the broad research challenges. The options extend it in a number of directions. They are strongly interdependent, building on specific aspects of Information Devices and Utility functionality to enhance our Applications, thus better demonstrating the latter's ability to amplify human intellect.

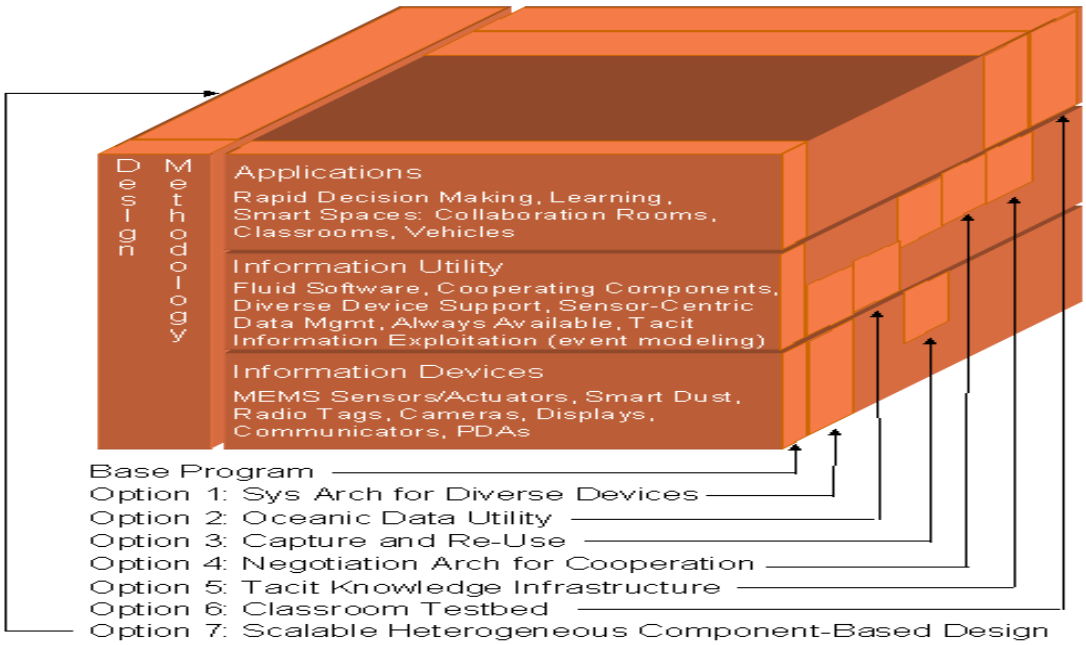


Figure 3-3. Overview of Project Endeavour [1]

3.3.4 Portalano (Washington University)

Computing and telecommunications are maturing, and the next century promises a shift away from technology-driven general-purpose devices. Instead, the focus will be on the needs of consumers: easy-to-use, low-maintenance, portable, ubiquitous, and ultra-reliable task-specific devices. Such devices, although not as limited by computational speed or communication bandwidth as their predecessors will instead be constrained by new limits on size, form-factor, and power consumption. Data generated by these devices will need to be injected into the Internet over a variety of wireless media. Data collected at the services will also have to find its way back to the devices,

wherever they may be and whenever they may be again connected to the network.

These are not simply the problems of ad-hoc networking and palm-sized PDAs. For computing to reach the broadest spectrum of our population, the entire infrastructure and the devices themselves must be as invisible as possible, requiring little or no configuration and performing reliably and predictably. To accomplish this requires a review of our basic assumptions regarding user interfaces and network transactions. It challenges us to develop entirely new models for distributed services. User interfaces will have to be based on multiple types of input ranging from the user's physical movement, location, data available from the network, and a variety of sensor data we can only begin to imagine today. Network topologies will be intermittent and services will have to be discovered independently of user guidance. The network fabric will have to provide computing and storage cycles to the data bundles. Data will need to find its own way from the user to the services and back, possibly replicating itself along the way to prevent data loss. We will require an open services architecture that not only permits users to have their data directed to the appropriate services but also allows the services to interconnect with each other.

The proposal of Project Portalano is to work in each of these areas to create a prototype of the future consumer computing landscape. We will deploy and study the elements of this vision [5].

User Interfaces: New modes of interaction such as user movement, proximity of devices, and embodied information presentation will augment the keyboard, pen, audio, and video interfaces we see today. Data fusion (combining the data gathered from location sensors, identification tags, and on-line databases) will be crucial in determining user intent rather than relying on user commands.

Network Infrastructure: The networking fabric must provide robust data transfer with replication and discovery as well as the ability to marshal computing resources at internal network nodes. The network must be data-centric in that transmission, routing, authentication, and resource reservation should be handled independently of the location and the media from which the user injected or will receive the data.

Distributed Services: Rather than abstract capabilities, emphasis must be placed on applications to which users can easily relate. These services will have to be more openly organised into horizontal layers rather than the vertically integrated monolithic services of today to better facilitate consumer choice. New development environments are needed to make services (with a palette of user interfaces) easier to design and deploy into the network and partition across our myriad computing devices.

3.3.5 Planet Blue (IBM)

The purpose of Project Planet Blue [4] - to creating a living laboratory - is to understand how people will interact with the emerging world of the wireless Internet. The initial focus is on the knowledge worker and how the knowledge worker interacts with other people, with information, and with organisations. A team of over 40 behavioural and computer scientists, across five of the IBM Research labs, is working to define user scenarios and to build compelling applications that will be deployed in studies with real users and within IBM Research itself. The applications will help drive requirements of the underlying infrastructure needed to support them. Planet Blue will help define the future of post-PC personal computing and drive our research in information access devices.

In order to deliver applications to different kinds of access devices, we will develop and use wireless networking and pervasive computing technologies. Two other critical areas of our focus are user interfaces, in order to get the end user experience right, and knowledge acquisition and management, in order to target relevant information to end users as precisely as possible. Applications focus on assisting individuals and teams.

The goal for Planet Blue, stated simply, is to create and deploy a technology-assisted immersive environment used by knowledge workers in their daily lives, in which individuals and teams can create, learn, use and share knowledge with few limitations or disruptions, regardless of physical location or context. We are trying to remove the technology from the consciousness of our users. We want computing to disappear into the background as well, leaving only the power of computation and global access - without the annoyances of techno-trivia. This means that we need to seamlessly

integrate a number of applications with information, and focus on making interaction with both as transparent as possible. As an example, a simple conversation relates to a number of different kinds of information, including calendaring and tasks. We must make the creation of these shared tasks trivial, and we must allow for appropriate access to a number of different information axes from different kinds of access devices, spanning small wireless devices and desktops.

3.4 Summary

In this chapter, the necessary background knowledge for understanding the rest of thesis is introduced.

- The term “legacy system” describes an old system which remains in operation within an organisation
- Software evolution is the process of adapting an existing software system to conform to an enhanced set of requirements. Software reengineering is software evolution performed in a systematic way.
- Wide Spectrum Language (WSL) has been developed for a number of years and has been used to build a general approach and a tool for addressing research issues such as program comprehension and reverse engineering using program transformation and abstraction techniques.
- A number of leading technological organisations are exploring pervasive computing.

Chapter 4

Proposed Approach

Objectives

- To summarise the rationale for the proposed approach, PerComE (A Framework-based Approach to Evolving Legacy System into Pervasive Computing Environment).
- To introduce proposed framework of PerComE approach.

4.1 Overview

The concept of pervasive computing now has grown far beyond its original intent. In fact, many applications can benefit from the pervasive infrastructure, including collaborative engineering, data exploration, high throughput computing and distributed supercomputing. For pervasive computing applications development, there is indeed a need to smoothly, seamlessly and dynamically integrate and deploy autonomous software.

Making existing applications run in a pervasive computing environment will increase resource utilisation and sharing. The ability of the legacy system to be used in pervasive computing environment will enrich the resources of the future pervasive computing environment and supply more working ability. This thesis focuses on establishing a general framework and methodology to assist with the evolution of legacy systems into pervasive computing environments.

As pervasive computing technology advances, the first step to come closer to realise the pervasive computing vision, is to create a list of requirements. Pervasive computing system has the same characteristics as other computing systems, though it has many more requirements than prior computing system, such as distributed system and mobile computing system:

- The ability to dynamically discover and compose software components in frequently changing environments
- The ability to support increasingly autonomous and invisible applications through the provision of rich context information that is gathered from a wide range of sources, interpreted, and disseminated in a scalable fashion to interested parties
- The ability to rapidly develop and deploy flexible software components that are adaptive and context-aware and, additionally, satisfy special requirements such as scalability and fault-tolerance
- The ability to integrate heterogeneous computing environments, which have differing communication protocols and services (such as discovery mechanisms), into coherent pervasive computing systems that enable the formation of dynamic interactions between components
- The ability to construct novel types of user interfaces that are universally available, regardless of the input and output capabilities of the available devices, that are sensitive to situation, and that are non-distracting

There are some major research challenges people must overcome before achieving these requirements.

- Open Standard: pervasive computing environment should be designed and implemented in an open and extensible manner, letting people combine components to form applications unforeseen at the time of their deployment. Technically, this implies obvious features such as open interfaces and support

for inter-component communication.

- Intelligence: In studying Weiser's scenarios of a pervasive computing world, people can clearly get the sense of some form of intelligence working on the user's behalf to coordinate the actions of components in the infrastructure.

- Applications integration: As the number of applications operating in a pervasive computing environment increases, coordination between these applications is needed. This coordination might range from traditional areas such as arbitrating screen usage to new challenges such as deciding which application may use the intensity of the light in a room to communicate with the user.

- Adaptation and contextual sensitivity: The environment in which a ubiquitous computing component functions is subject to change. Such changes might be prompted by variations in resource availability as a result of failures or the deployment of new services or by variations in patterns of usage or mobility. The importance of adaptation is well understood in the field of mobile computing. However, it is significantly more complicated in pervasive computing systems, where there is a need to respond to a much larger set of contextual triggers. More importantly, we might also have to substantially reconfigure applications involving multiple components. The need to manage these configuration changes in ad hoc ubiquitous environments poses significant problems.

So, there is a need for new approaches to fulfilling above requirement and challenges. An agent-based service-oriented integration approach is proposed in this thesis. First, reverse engineering techniques are used for program comprehension and design recovery. Then the legacy software systems are decomposed into a hierarchy of subsystems by defining relationships between the entities of the underlying paradigm of the legacy system. Next, pervasive services are created by wrapping subsystems and/or components. Finally, Multi-agent system is used to allocate and integrate pervasive services into pervasive computing environment.

4.2 Agent-based Service-oriented Approach

The proposed agent-based Web Services evolution approach provides a standard mechanism allowing Web services and agents techniques to be brought together in re-engineering legacy system by providing migration paths to allow smooth evolution. With this capability, agent technology is provided with which a great number of resources and services can be accessed. As a consequence, it opens up a new research area in pervasive computing environment and makes it one step closer to achieving Weiser's vision. The propsed approach consists of two major steps: component based pervasive services identification, and agent-based system integration and deployment.

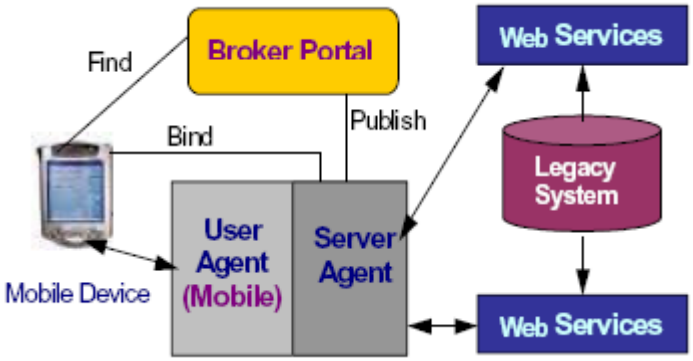


Figure 4-1. Agent-based Service-oriented Approach

4.2.1 Component-based Pervasive Services Identification

Component based development is the industrialisation of the software development process based on the assembly of prefabricated software components. Two basic ideas underlie component based development. Firstly, application development can be significantly improved if applications can be quickly assembled from prefabricated software components. Secondly, an increasingly large collection of interoperable software components will be made available to developers in both general and specialist catalogs.

Software Component could be defined as: A software component is a unit of composition with contractually specified interfaces and explicit context dependencies

only. A software component can be deployed independently and is subject to composition by third parties. Component-based system design is the major approach in the construction of pervasive computing system. As for system evolution, useful components can be extracted from existing system using the reengineering technologies. After the legacy system components identification, these components could be migrated as services in the pervasive environment. Only if a legacy system have some of the following characteristics, it may be suitable for a component-based and service-oriented software reengineering process.

- a legacy system or part of it needs to be migrate into a distributed environment and can be wrapped and exposed as software services;
- a legacy system have some reusable and reliable functionalities embedded with valuable business logic;
- reusable components extracted from a legacy system are fairly maintainable compared to maintain the whole legacy system;
- the functionalities within a legacy system are significant and independent to be exposed in the service-oriented environment from the requirements pointof view;
- some components of the target system run on different platforms and vendor products;
- some legacy components will be replaced gradually without affecting service consumer;
- the target system will operate over Internet where reliability and speed cannot be guaranteed.

Today, Web services are emerging as the new “standard” architectural style. This new architectural style and the software lifecycle it implies are extremely attractive because they can effectively address the demands for short development cycles, distributed development and global user base, at the same time. The new applications

developed in this style effectively reuse existing software assets to provide new complex value-added services, fast [107]. In order for this vision of effective reuse-based development to become a reality, a wide base of available Web services from existing systems is required. For Pervasive Service migration, the approach is based on XML representation and transformation. Once a software component has been extracted from a legacy system, or has been built as a new component, the legacy components are wrapped as XML components which could be further used in the Pervasive services environment.

4.2.2 Agent Based Pervasive Services Integration

Through agent-based integration, a migration path to allow smooth evolution of agent-based Web services with interaction abilities is provided. The Administrator works as an agent server and gets Web Services invocations. It looks up the service repository and uses the corresponding Web Services. Meanwhile, it is essential for Administrator to know the Web Services execution results and log the invoking history. The agent is responsible for mapping between the various actors within the model and functions independently of the system. The agent instance executes the appropriate plan or task according to Web Service invoking parameters and depending on the service provider's circumstances. The agent would interact with provided Web Services autonomously. The improper Web service invocation could result in that nothing can be invocated by Administrator. The proposed method can roughly be divided into following steps:

1. The legacy system is re-engineered into Agent-based Web services. Meanwhile, each target Web service can have at least one "accompanying agent" or "shadow agent" and this Web service can be accessed by the agent accordingly.

2. "Accompanying agent" can be split into two agents. One is stationary Agent named Server Agent and one is the mobile agent named User Agent. The Information of Server Agent will then be registered into the Broker Portal, which is designed for Mobile Device to browse.

3. After the mobile device connects and authenticates to the Broker Portal, The user enters a client query to perform a particular task by executing a client query request. When the Broker Portal receives the requests, it will inform the Agent server with a message. As the result, the Server Agent will migrate the split mobile agent to the Mobile Device.

4. With mobile user agent, the mobile device will execute the real task and save the user profile into the user agent. If at any point in time, the client experiences a network disconnection, crash or a move to a different location and possibly a different type of network, then server agent is able to recover from the service that the user had previously accessed, and continue using it.

5. After the user has finished the task with server agent, a logout request is sent to Broker Portal.

Advantages of the proposed approach:

- Easy support for mobile agents with a fixed home base(Server Agent), which mobile agent will be a thin agent and can migrate easily.

- Agent can migrate and the save the execution state, which is suitable for pervasive computing environment.

- Broker Portal can provide the unified services.

4.3 Summary

In this chapter, a unified software evolution approach (An Agent-based Service-oriented Approach to Evolving Legacy Software Systems into a Pervasive Computing Environment), is proposed.

- An architectural framework in evolutionary pervasive computing system design is given. Web Services and agent techniques can be brought together in re-engineering the legacy system by providing migration paths to allow smooth

evolution.

- The approach consists of two major steps: Component-based Services Identification, and Agent-based Pervasive service Integration and Deployment.
- The research in this direction is quite recent and far from resulting into a completely automatic transformation process.

Chapter 5
Component Based Service Identification for Pervasive Computing

Objectives

- To identify, classify and define the component using in Pervasive Computing environment.
- To discuss the technologies in service-oriented component mining in legacy systems
- To introduce the rules for service-oriented component extraction.

This chapter defines a reengineering and migration method that identifies services from legacy system for use in Pervasive computing environment. Such a reengineering framework is composed of legacy system decomposition, component identification, components composition and service environment integration. Various services will be created dynamically at different time for different user's needs. These services are transparent to users. Users just use services they want regardless where these services come from and how many individual services integrated together.

5.1 Component-based System Decomposition

Component based software engineering is a process that aims to design and construct software systems using reusable software components. The component paradigm starts with the assertion of an assembly-oriented view of software engineering, building

software applications by wiring together the ports and connectors of a set of pre-fabricated parts (components) within a component context. Decomposing a program entails the identification and the reorganisation of different program components.

If the component based approach is used for reengineering legacy system, a component mining process will be performed. Reverse engineering techniques of static program slicing and hierarchical agglomerative clustering are applied in this component mining process. The program slicing technique is used to decompose system, understand program, eliminate dead code and make selected code segments function independently by component interface parameters determination and deep source code comprehension and analysis. The software clustering technique is used to group large amounts of entities in a dataset and capture reusable legacy code segments into clusters according to their relationship and similarity from legacy systems, and create a hierarchical structure of these reusable legacy code segments. These reusable legacy code segments are independent, self-contained, coarse-grained and loose-coupling. Mostly, legacy systems are huge and complex. Using components based development approach to evolving legacy systems is less risky and highly transport and it will also bring more flexibility, expansibility, reusability and reliability. Also, this approach is low cost and easy to implement.

5.1.1 WSL Based Program Analysis

WSL is a multi-layered wide spectrum language with sound formal semantics. Both the object-oriented and procedural systems can be transformed into WSL. The features of WSL including the tool developed for WSL transformation are as follows:

- a small, traceable kernel language with very precise and thorough formal semantics;
- a set of transformation rules for restructuring and simplification;
- object-extraction rules to enable transferring legacy procedural programs to object oriented programs;

- abstraction rules for crossing levels of abstraction in a stepwise manner and abstraction patterns as a means of describing current abstraction situations and acquiring expert observations of the target system, and then applying these observations in further abstraction;

- an interactive, semi-automatic tool support, thereby making good use of human expert knowledge about the software and its domain;

To translate source code into WSL code has been researched and discussed clearly in [135]. Once the source program has been captured in WSL, there are a large number of restructuring and simplifying program transformations that can be applied automatically to clean up the code, unscramble the structure, and delete redundant code. The result is a structured program consisting of a hierarchy of single-entry, single-exit procedures.

5.1.2 Program Slicing

Program slicing is a method for automatically decomposing a program by analysing its control and data flow. With the help of slicing technology, legacy systems can be divided into some concerned program parts and discard the useless assets. This section deals with software systems composed of programs each of which may comprise all types of components and proposes a technique to decompose them. The proposed approach exploits static analysis and program slicing techniques to identify components to be used in pervasive computing environment. Traditional, it is an established technique for reverse engineering.

There are two main approaches to slicing: The original slicing technique from Weiser is based on data flow and control flow analysis; the other approach is based on Program Dependence Graphs (PDG). The control flow analysis is adopted in the proposed approach for program representation and static analysis of programs at the intraprocedural and interprocedural levels.

A Control Flow Graph (CFG) is a representation, using graph notation, of all paths that might be traversed through a program during its execution. Directed edges are used to represent jumps in the control flow. In most presentations, there are two specially

designated blocks: the entry block, through which control enters into the flow graph, and the exit block, through which all control flow leaves. A CFG for program P is a graph in which each node is associated with a statement from P and the edges represent the flow of control in P. Let V be the set of variables in P. With each node n associates two sets: REF(n), the set of variables whose values are referenced at n, and DEF(n), the set of variables whose values are defined at n.

Computing a slice from a control flow graph is a two step process: at the first step, requisite data flow information is computed and then this information is used to extract the slice. The data flow information is the set of relevant variables at each node n. For the slice with respect to <*s, v*>, the relevant set for each node contains the variables whose values affect the computation of v at s. The second step identifies the statements of the slice, which include all nodes (statements) *n* that assign to a variable relevant at n and the slice taken with respect to any predicate node that directly controls *n*'s execution. In the proposed approach, rules of slicing and chopping are based on dividing the system into independent, stable and high reusable segments. Each part can be evolved and served as component resources, which can be connected with each other flexibly and effectively and can be integrated as services.

5.1.3 Code Segment Extraction

Programming Slicing algorithm is defined to extract component from legacy code. In the proposed approach, backward, static, inter-procedural slicing algorithm is used to generate slices according to predefined slicing criteria. Then these slices are united according to the following algorithm. It is very useful in software reuse and reengineering.

Taking the variables occurring in the key statement as a slicing criterion, a backward slice is formed that incorporates all statements relevant to the computation of these variables. More formally, the proposed slicing algorithm is a function which takes key statements, *{s1, s2, ..., sm}*, and code segments, *{P1, P2, ..., Pn}*, and produces an executable sub-component, which contains key statements *{s1, s2, ..., sm}* according to code segments *{P1, P2, ..., Pn}*. The description of the algorithm uses the following

functions:

- *BackwardSlice(P, s)* returns a backward slice using provided slicing criterion
- *RemoveDuplicates(slices)* returns a set of distinct slices
- *Validate (slice, Functions)* returns true if the slice implements some of the functions

This last function requires the user interaction to select the backward slice which implements the functional abstraction among the candidates obtained with a different statement in the slicing criterion. This is a concept validation task because code segments are filtered through a given human-oriented concept. Although the process requires a frequent validation to choose the right slice among the candidates, the user is asked to read small similar pieces of code compared to the large amount of code necessary to identify the statements of the expected function.

5.1.4 Code Segment Clustering

Components are good candidates for modelling units of functions because they encapsulate attributes and methods to act as independent entities. In order to group the sliced legacy code segments and to create a hierarchical structure of the system, the cluster analysis is applied. Software clustering technique groups entities into clusters according to their relationship and similarity. It can be applied to the captured reusable legacy code segments to form independent, self-contained, coarse-grained and loose-coupling component.

Current cluster analysis offers a wide range of techniques for identifying underlying structures in large sets of objects and revealing relationships between objects or the classes of objects. Most clustering techniques utilise certain criteria to decompose a system into a set of meaningful modular clusters. Such criteria attempt to achieve a cluster with low coupling, high cohesion, interface minimisation and sharing of neighbour resources. The proposed approach has preferred hierarchical algorithms for component identification in legacy systems because they support a manual refinement of the clustering granularity and they are more suitable for further restructuring by their

hierarchical structure. Hierarchical algorithms do not produce a single partition of the system. Their output is rather a tree, with the root consisting of one cluster enclosing all entities, and the leaves consisting of singleton clusters. After selecting a clustering tree level, the user can move upward or downward in the tree, if the clusters at the selected level are too specific or too general.

5.1.4.1 Similarity between Entities

When two clusters are joined or one cluster is split up in two smaller clusters, the similarities between the newly formed cluster(s) and the previously existing ones have to be calculated. There are many common methods for defining similarity between clusters includes the single-link method, the complete-link method, ward's method, etc. [73].

In the single link method, the similarity between two clusters C1; C2 is the maximum of the similarity between a pair d1; d2. In the complete link method, the similarity between two clusters C1; C2 is the minimum of the similarity between a pair d1; d2. In ward's method, the pair of clusters that are considered to be closest together among all clusters (and hence are merged in a step of a hierarchical clustering algorithm) is the pair whose merger minimises a certain sum of squares error based on distances from centroids of the clusters. The complete link method is usually favoured in reengineering applications, because it tends to produce smaller and more tightly-linked clusters.

An entity could represent a subsystem, a directory, a file, a class, a function, a data structure, a variable, and so on. A resemblance coefficient for a given pair of entities indicates the degree of similarity or dissimilarity between these two entities, depending on the way in which the data is represented. A resemblance coefficient could be qualitative or quantitative. Whether qualitative or quantitative data should be chosen depends on the applications and attributes.

For the legacy systems which are written by procedure-based code, the entities are defined as functions. In order to apply the clustering technique, the cluster technique is tailored based on function-function inter-connections. In this case, the data set

represents the interdependencies or interconnections which are indicated by function calling relationships. For objected-oriented programs, the coupling information could be class-attribute, class-method, method-method, the inheritance among classes, or a shared feature.

5.1.4.2 Similarity Calculation

When a similarity measure was used that can be applied to sets of entities these similarities can be computed from the original data. To ascertain the similarity between two entities, the proportion of relevant matches between the two entities is calculated. There are different methods of counting relevant matches and there are many algorithms to calculate the similarity or resemblance coefficient.

For software reengineering purposes, the proposed approach chooses algorithms which impose a structure satisfying the constraints that a good modularisation or componentisation should obey. In input data matrix, the 1 entries show the corresponding functions that are interconnected. The matrix is symmetrical and 1 entries are used in the main diagonal. The numeric values (1 or 0 entries), which show the number of interconnections among functions, are obtained from a static analysis of the source code. The number of function calls is based on syntax analysis, however it does not reflect the actual number of invocations of those functions. Dynamic information may be more insightful, but it is difficult to obtain and highly dependent on run time behaviour.

5.1.4.3 Clustering Algorithm

The clustering algorithm is a sequence of operations that incrementally groups similar entities into clusters. The sequence begins with each entity in a separate cluster. At each step, the nearest two clusters are combined into a higher-level cluster. The distance between any two clusters is taken to be the average of all distances between pairs of objects. Then, dendrograms are adopted to demonstrate the clustering process and the proximity between entities. It presents a hierarchical view of the legacy system, which consists of different levels of abstraction from source code to subsystems.

The hierarchical agglomerative clustering approach can be described as follows:

1. Start with a set of singleton clusters, each containing one document and initially unmarked.

2. Repeat the following steps iteratively until there is only one cluster left unmarked.

    a. Identify the two most similar unmarked clusters, and mark them.

    b. Form a (new, unmarked) parent for these clusters by merging them together into a single cluster.

    c. Updates the similarities between the clusters.

In order to extract a functional component from legacy code, a cutting point must be figured out in the dendrogram. This cutting point affects the number and quality of the components derived from the dendrogram. It is an elementary factor for determining the granularity of extracted components. The clustering at the level of the cut point is the resulting clustering. So far, the cutting point is determined with architects' participation and user requirements. Other existed and recovered design information facilitates this decision-making process. This clustering method together with human supervision restructures legacy systems into coarse-grained and loose-coupling components.

5.2 Component Wrapping as Web Services

After the legacy system components identification, these components have to be migrated for deploying in the pervasive computing environment. Some extracted components should be wrapped, which the goal of the wrapping process is to provide the component extracted from the legacy code with a WSDL interface. The technique used is to transform each entry into a method and to transform each parameter into an XML data element. The data structures will become complex elements with one or more sub-elements. The methods will have their arguments and results as references to the data element descriptions. Both the methods and the parameters will be built into an

XML schema with a SOAP framework.

If the reusable legacy component is wrapped as a Web service straightforward, the following activities are performed. Figure 5-1 shows the wrapping process. Service-oriented components, together with all of it's sub-functions, are wrapped according to the result of service-oriented analysis.

A service interface is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any conditions implied by those messages. Furthermore, a service must have sufficient service descriptions which associated with the service to enable its use by other parties. Ideally, a service description will give sufficient information so that an automatic agent may not only use the service but also decide if the service is appropriate; that in turn implies a description of the semantics of the service. Because Web services are network accessible interfaces to application functionality, built using standard Internet technologies, a Web service interface is applicable to most software services. If the target service is to provide as a Web service, a rigorous definition of the functionality of the service is provided in Web Services Description Language (WSDL). WSDL also describes the operations that a service can support, and the parameters each operation accepts and returns.

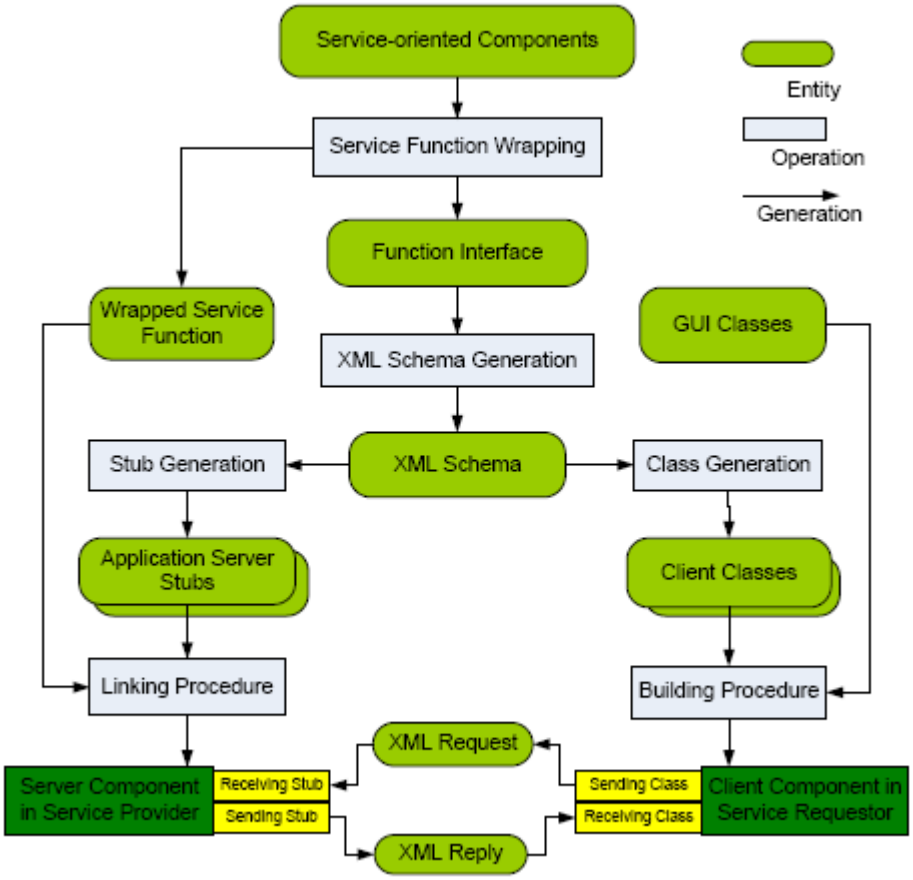


Figure 5-1. Web Services Wrapping

5.2.1 Components Representation Using XML

In the proposed approach, the XML plays a great important role. XML is used to describe the data structure and exchange information. Extensible Stylesheet Language (XSL) allows building legacy codes which are insulated from changing the external format of both the legacy system and the calling application. XSL is also used to manipulate XML from one structure into another. At last, XML schemas are employed to encapsulate and integrate legacy components.

Web interfaces have been used increasingly for applications that have been thought of previously as traditional, single-user applications. Extensible Stylesheet Language (XSL) is introduced to achieve the Web interface transformation and to provide a better control over legacy systems. Extensible Stylesheet Language Transformations (XSLT) is an

XML-based language which is used for the transformation of XML documents. It can manipulate XML from one structure into another. Therefore, it can be implemented for transforming XML into HTML or other XML document structures.

An XSLT stylesheet can make use of information from a schema, and an XSLT transformation can take place in the absence of a DTD. The XSLT processor has access to the type information associated with individual nodes, not merely to the untyped text. The important roles of XSLT is to add styling information to an XML source document, by transforming it into a document consisting of XSL formatting objects or into another presentation-oriented format such as HTML, XHTML. However, XSLT is used for a wide range of transformation tasks, not exclusively for formatting and presentation applications.

Many Web-based systems can dynamically produce client side HTML pages from XML documents via XSLT and is used to format or transform XML content from one form to another. The benefit of the approach is that without any efforts to duplicate the legacy code, all features from legacy systems are inherited. Furthermore, they can easily be integrated with each other to perform dynamic services in a pervasive computing environment.

XSL can be used not only for formatting an incoming document structure but also for formatting the data returned from the legacy system into a usable structure for the function or application using API. XSLT processing often begins by reading a serialised XML input document into the source tree and ends by writing the result tree to an output document. The output document may be XML, but can be HTML, plain text or any other format that the XSLT processor is capable of producing. The transformation is achieved by a set of template rules. A template rule associates a pattern, which matches nodes in the source document with a sequence constructor.

5.2.2 Service Packing Process

The service packing approach is based on XML representation and transformation. Once a software component has been extracted from a legacy system, or has been built

as a new component, its interface can be extracted and represented in XML. The XML representation is not only finished by the component wrap, but also with the sources code analysis included, such as the using of DTD and XSLT. At last, the legacy components are wrapped as XML components which could be further used in the pervasive computing environment.

XML Schema Generation: Following the wrapping of the service-oriented component and the creation of new functional interfaces, the next step is to transform those interfaces into an XML schema while, at the same time, generating server stubs to process XML messages based on that schema. When the functions are wrapped, their interfaces are created in the original language.

Server Stub Generation: The parsers for processing the incoming XML messages from the clients and for organising the XML messages back to the clients are generated in this step from the XML schemas. For the sake of compatibility with the runtime system the target functions are running in, the parser stubs are created in the same language as that of the functional modules.

Client Class Generation: The same XML schemas, which are the basis for the generation of the server stubs are also used to generate the client classes. For each web service, two functions are generated, one for creating the request to the server function and one for interpreting the result returning from the server.

Web Service Binding: The application server functions are linked together with their stubs into individual dynamic link load modules to be made available on the host. They can be invoked from any web browser anywhere in the internet. The invoking client component need only include the classes for sending and receiving the XML interfaces to the server functions.

5.3 From Web Service to Pervasive Service

Pervasive services is defined by combining Web services and Pervasive computing to perform a seamless information processing system across distributed, heterogeneous,

dynamic virtual organisations. It is basically the Web services with improved characteristics and services. Web services are the technology for Internet based applications with loosely coupled clients and servers. However, the implementations of Web services are typically stateless. Compared to this, Pervasive services have evolved to make it possible to dynamically share and coordinate heterogeneous service resources.

The pervasive service is an extension of the current Web Service in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation. It arises with the parallel development of Web services, Semantic Web and pervasive Computing, in particular the endeavour of applying the former on the latter. While both Semantic Web and pervasive computing focus on the operating in global distributed and changeable environment. The Semantic Web works on small number of hosts and it only provides static metadata.

For pervasive service, stateful and semantic are eminent features. In the pervasive computing environment, service oriented architecture provides an interface for individual services to bind up as a new service, and break down integrated services dynamically. Resources management picks necessary services and creates a new service for each special requirement. With regard to Service oriented architecture, Semantic Web supports persisting Web services and stateless Web services. It provides dynamic metadata and supports transient dynamic and stateful pervasive services.

With the support of stateful resources can remember what have been done from one invocation to another. The term state is vague and can encompass many different aspects of a computer system, from the value stored in a specific database record to the seek time or even temperature of the disk drive. A stateful resource can be defined as resources which have following features: have a specific set of state data expressible as an XML document, have a well-defined lifecycle and to be known by one or more services.

Migrating Web Services to Pervasive Services aims to assign semantics to the reengineered Web Services. In the proposed approach, the RDF data models which are based on a specific XML mark language RDF/XML are employed to provide a simple

and elegant frame for describing the properties of the reusable legacy system resources. Ontology will be introduced into the component mining approach. RDF/XML representation and RDF-schema description are key techniques to reuse recovered Web Services in semantic pervasive services.

5.3.1 Describing Pervasive Resources Metadata

This section focuses on how to retrieve the useful resources from legacy systems and represents the resources based on the Semantic standards. The Resources Description Framework (RDF) is a language for representing information about resources. To present properties, resources and statement, W3C has defined a concrete syntax based on the Extensible Markup Language (XML). A specific XML mark language RDF/XML has been defined by RDF for representing RDF information and for exchanging it between machines. It focuses on describing metadata on the World Wide Web (WWW). For legacy systems, heterogeneous resources have been forced to follow the RDF to produce a very general model and syntax, which can be examined without referring to the legacy system at all. Rather than only display to people, RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning.

Four components have been defined in a RDF Data Model []. (a) Resources: a resource is anything which can be named with a Universal Resource Identifier (URI). (b) Literal: Atomic values, such as integers or strings. (c) Properties: a property is a specific aspect, characteristic, attribute, or relation used to describe a resource. (d) Statements: statement is an ordered triple of (predicate, subject object), where predicate is a property, subject is a resource, and object is a literal or a resource. The concerned resources are extracted and packaged as components which are ready to use in the future extension. Figure 5-4 shows the RDF/XML for conference information.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 <rdf:Description about="grid://www.cs.fiu.edu/IRI05/>
   <g:conference>
    <g:Description about="grid://schemas/conderence">
     <g:name>The 2005 IEEE International Conference on Information Reuse and Integration</u:name>
     <g:date>Aug 15-17, 2005</u:date>
     <g:venue>Las Vegas-Nevada-USA</u:venue>
     <g:sponsors>IEEE Systems, Man and Cybernetics Society</u:sponsors>
    </g:Description>
   </g:conference>
 </rdf:Description>
</rdf:RDF>
```

Figure 5-2. RDF/XML Representation for Conference Information

<?xml version="1.0"?> is the XML declaration, which indicates that the following content is XML, and what version of XML it is. <rdf:RDF> element indicates that the following XML content is intended to represent RDF. An XML namespace declaration is defined on the same line, presented as an xmlns attributed of the rdf:RDF start-tag. This declaration specifies that all tags in this content prefixed with rdf: are parts of the namespace identified by the URIref http://www. cs.fiu.edu/IRI05/. Beginning with the string http://www.cs.fiu.edu/IRI05/, URIrefs are used for terms from the RDF vocabulary. The ‘Description’ element relates to a resources, whose URI is provided in the ‘about’ attribute. Then the RDF/XML representations are provided for the specific statement. In the conferences representation, the literal values are: ‘The 2005 IEEE International Conference on Information Reuse and Integration’, ‘Aug 15-17, 2005’, ‘LasVegas-Nevada-USA’ and ‘IEEE Systems, Man and Cybernetics Society’.

In order to reusing the reengineered pervasive computing resources, a mechanism for declaring the properties and for defining the relationships between these properties and other resources must be provided. RDF itself provides no means for defining such application specific classes and properties. Instead, such classes and properties are described as an RDF vocabulary. Using extensions to RDF provide by the RDF Vocabulary Description Language 1.0, referred as RDF Schema.

Because RDF-Schema is itself an instance of RDF, the syntax is also defined by the RDF XML syntax. In RDF Schema, core classes are defined using rdfs: resources, rdf: property and rdfs: class. Properties are described using the RDF class rdf: property, and

the RDF Schema properties rdfs: domain, rdfs: range, and rdfs: subPropertyof.

5.3.2 Semantic Pervasive Computing Framework

The system evolution in pervasive computing environment includes many aspects. Pervasive computing system also has the same characteristics as other computing systems, though it has many more requirements than prior computing system, such as distributed system and mobile computing system.

The evolution to pervasive computing is characterised by a marked increase in mobility, embeddedness, information technology integration, device heterogeneity and the communication capability from an infrastructure point of view. These basic characteristics and other attributes—primarily derivatives in nature—such as adaptivity, smartness (intelligent infrastructure), localised scalability, uneven conditioning and context sensitivity are the basis for the evolution of pervasive infrastructure.

Semantic pervasive services, which may include the following aspects: Expose the meaning of pervasive services, resources and entities by assertions in a common RDF data model. Publish and share consensually agreed ontologies in web ontology language OWL and query, filter, integrate and aggregate the metadata in RDF Data Query Language (RDQL).

Using Web Services technologies to implement a distributed system doesn't magically turn a distributed object architecture into an SOA. Nor are Web services technologies necessarily the best choice for implementing SOAs. Still, we should mention that Web services are increasingly becoming an adequate technology for the partial implementation of features of a service oriented architecture. The problem that we are addressing is how to provide the services from and perform service invocation and roaming that requires minimal infrastructure changes in pervasive computing environment.

Service interoperability is an important requirement to achieve so that services can interact with each other and with clients in a uniform manner, related to this, is the easy integration of services. It also allows us compose basic services to form more

sophisticated services and underpins the ability to map common service semantic behaviour seamlessly onto native platform facilities.

Knowledge and Semantic Web technologies are evolving the Web Services towards the Semantic pervasive services to facilitate knowledge reuse and collaboration within a community of practice. Retargeting to Semantic pervasive services builds up new systems by integration of components in the reusable library. It involves activities at the turning point between reverse engineering and forward engineering. Its retargeting involves functional restructuring and the start of forward engineering.

5.4 Summary

- This chapter describes an approach to identify concerned resources from legacy systems for designing services which are used in pervasive computing environment.
- The static program slicing techniques are applied to decompose program as a preliminary step for the migration of legacy systems. The control flow graph is adopted in the proposed approach for program representation and static analysis.
- Software clustering techniques are applied to capture independent legacy code segments. An improved agglomerative hierarchical clustering method is proposed to extract independent services from legacy code.
- Service-oriented components generated by previous component mining techniques are reused via wrappers (component adaption). Wrapping technology is used to remove mismatches between the interface exported by a software artifact and the interfaces required by current integration practices.

Chapter 6

Agent Based Service Integration for Pervasive Computing

Objectives

- To design a Pervasive service enabled integration architecture.
- To build agent based middleware to integrate the identified services to a heavily heterogeneous pervasive computing environment.

From above chapter's description, we can see that, only with services and Web services, two problems are not solved. One problem is that web services are focused on B2B application while pervasive computing asks for B2C application for the end users. Another problem is that service paradigm does not address mobility issues to perform service roaming and handle user migrations. Agent-based approach will extend these limitations to B2C application and utilise mobility of agent to support the migration.

Since Web Service definitions can be mapped to any implementation language, platform, object model, or messaging system. As long as both the sender and receiver agree on the service description, (e.g. WSDL file), the implementations behind the Web services can be anything. In practical developments,. The need for some degree of autonomy, to enable components to respond dynamically to changing circumstances while trying to achieve over-arching objectives, is seen by many as fundamental. Because of the autonomy and intelligent nature of agents, agent-oriented techniques are a design metaphor for Web Services, where agents are needed both to provide services, and to make best use of the resources available. The agent instance executes the

appropriate plan or task according to Web service invoking parameters and depending on the service provider's circumstances. The agent would interact with provided Web services autonomously.

Developing software agents for pervasive computing is an interdisciplinary task demanding expertise in such areas as agent technology, intelligent environment, control network, and artificial intelligence. Multi-agent systems have been adopted to build intelligent environment in recent years. Multi-agent systems present a viable solution to handling the complexity of such a dynamic environment. Experiments have been carried on to gather expertise from both pervasive computing and agent technology society. While previous research has made significant advance in some aspects, most of them did not focus on providing a satisfactory system or model for learning individual preferences, which is the core issue of personalisation in pervasive computing environment. The concept of pervasive computing is a paradigm shift from a traditional computing point of view. So pervasive computing is characterised by a marked increase in mobility, embeddedness, information technology integration, device heterogeneity and the communication capability from an infrastructure point of view. These basic characteristics and other attributes - primarily derivatives in nature - such as adaptivity, smartness (intelligent infrastructure), localised scalability, uneven conditioning and context sensitivity are the basis for the pervasive infrastructure. This chapter introduces what agent integrated approach can provide to achieve this goal by:

- development of ontology based agent knowledge structures
- policy management and development of personal profiling mechanisms
- interface with low level control system
- design of intelligent personalised agents

6.1 Agent-based Integration Architecture

Pervasive computing will be a fertile source of challenging research problems in computer systems for many years to come. There are some major research challenges in

pervasive computing environment, such as software architecture, networking challenge, component interaction, user interface integration, e.g.. There are more and more users that carry mobile devices such as PDAs, cell phones and other portable devices. As a result, existing services and application need to be extended towards these mobile users, so that they can access services and data anywhere at anytime. The challenge our project aims to address is how to deliver these services and applications in a transparent, and it is vital for that pervasive computing can become more widespread.

We propose a form of architecture (see Figure 6-1) for Agent-based Integration, which provides a common generic platform capable of agents to execute and give consumers states and execution results of Web services. The proposed architecture consists of a Central Agent, Local Agent, Personal Agent, Control Agent, Policy Repository, Mapping Repository and Web services interfaces.

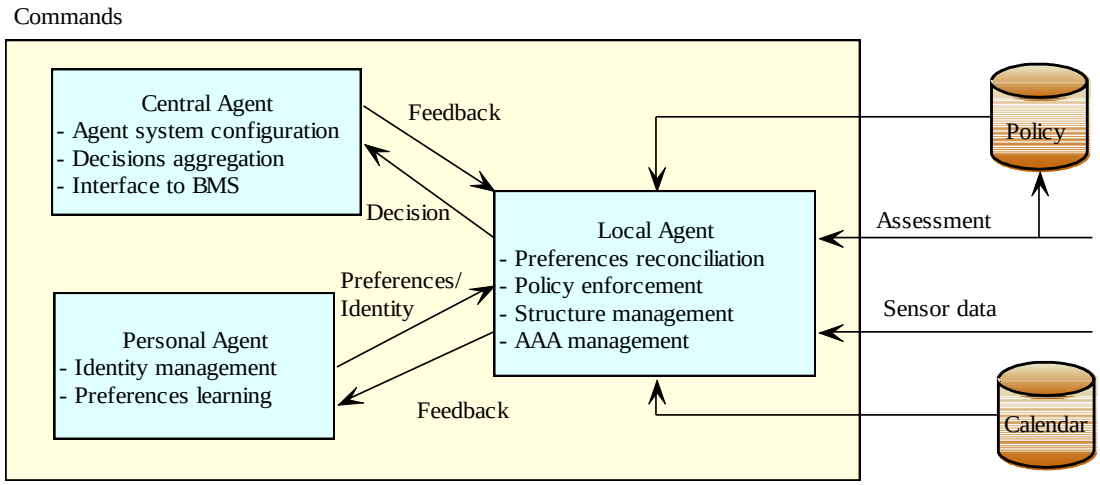


Figure 6-1. Agent-based Integration Architecture

There are four types of agents in:

- Personal agent: acts as an assistant that manages user (occupant) profile, observes the work environment, records user's behaviour, forwards operation requests, learn preferences and presents feedback from other agents to its user (occupant).

- Local agent: plays a central role. It acts as a mediator, policy enforcer and

information provider. It reconciles contending preferences from different users, enforces policies that constrain the possible settings for environment parameters and responds to environmental state change.

- Central agent: has two major functions: decision aggregation and interface to internal/external services required by other agents. The typical services provided by central agent include agent system configuration and interface to BMS.

- Monitor&Control Agent: enforces the operation request given by the user, reads and processes sensor data, and achieves an environmental state according to decisions made by the Local Agent.

The Central Agent works as an agent server and gets consumers' Web Services invocations. It looks up the Mapping Repository to get the corresponding local agents to create the corresponding Web Services. Furthermore, it is essential for Central Agent to know the Web Services execution results and log the invoking history. The local agent is responsible for mapping between the various actors within the model and functions independently of the system. The Personal Agent executes the appropriate plan or task according to Web Service invoking parameters and depending on the service provider's circumstances. The agent would interact with provided Web Services autonomously. The improper Web service invocation could result in that nothing can be invocated by Central Agent.

6.2 User Scenarios and Use Cases

User scenarios are presented to help develop a motivation for pervasive computing and to illustrate the concepts discussed. A typical scenario is that of a user who wished to access pervasive computing environment with his PDA or cell phones or interact with environment with his ID.

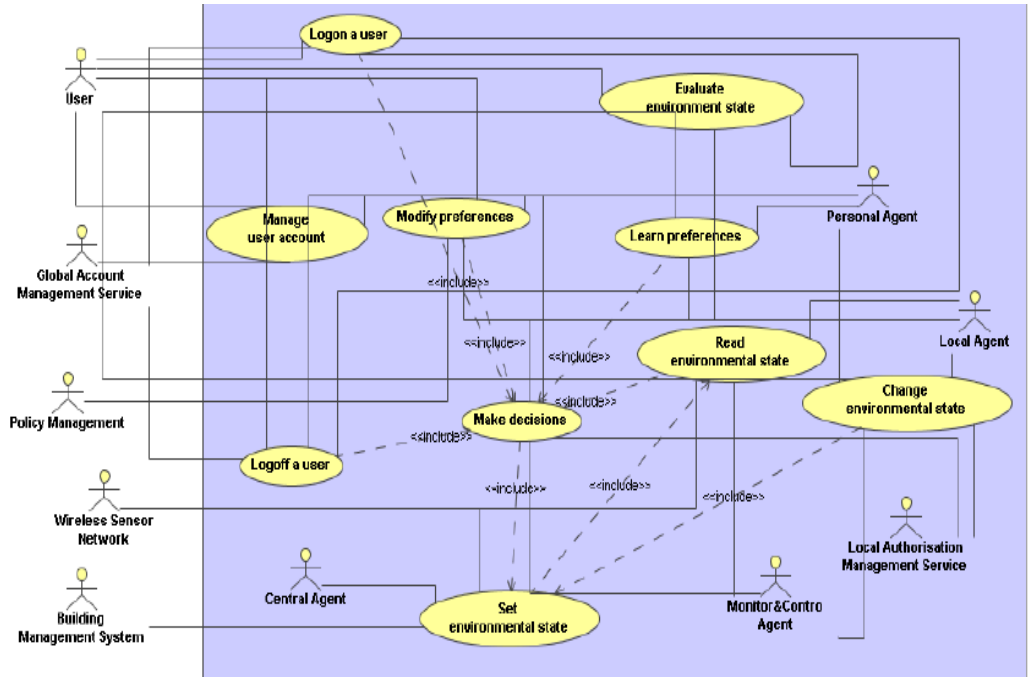


Figure 6-2. Use Cases

The use cases for the above scenario can be specified as use cases shown in Figure 6-2, which provides a common generic user requirements.

Use Case Section	Comment
Level	"user-goal" or "sub-function"
Primary Actor	Calls on the system to deliver its services
Stakeholders and Interests	Who cares about this use case, and what do they want
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling the reader
Main Success Scenario	A typical, unconditional happy path scenario of success
Extensions	Alternate scenarios of success or failure
Technology and Data Variations List	Varying I/O methods and data formats

Figure 6-3. Use Cases Section Descriptions

The labels, *a, *b …, are used to to describe an extension condition as possible during any (or at least most) steps, or a technology and data variation as possible for all use cases.

6.2.1 Technology and Data Variations List for All Use Cases

The labels, *a, *b …, are used to describe an extension condition as possible during any (or at least most) steps, or a technology and data variation as possible for all use cases.

*a. A Personal Agent stores user profile, preferences and dynamic rule set. A Personal Agent can be implemented by the following equipments:

1) Smart-card plus a shared networked PC.

2) Smart-card plus a personal mobile device (PDA).

3) A shared networked PC.

*b. The implementation of actuation depends on that of Personal Agents.

1) If Personal Agents do not have remote control over actuators, users will need to operate building facilities directly and notify their Personal Agents of their actions.

2) If Personal Agents have remote control over actuators, users will be required to operate building facilities via their Personal Agents so that their preferences can be learned.

*c. Special devices are used to detect a user's location to dynamically associate sensors with them.

*d. Sensors are associated with users according to their "most likely" locations.

6.2.2 User Login

Level

User-goal

Primary Actor

Local Agent

Stakeholders and Interests

- Local Agent: verifies the user, create a new session, retrieve user's preferences, and update environmental state.
- Personal Agent: identifies the user to the Local Agent, retrieve Zone information, and set current set of preferences, rule base and learning strategy.
- Global Account Management Service (GAMS): receives authentication request and provide general authorisation details to the Local Agent.
- User: triggers logon event.

Preconditions

The system is limited to recognised users with various authorisations for different devices in different rooms, and so the user needs to have appropriate proof of identity and related permissions to have his/her preferences learned in the Zone.

Success Guarantee

The user is authenticated and properly authorised. A new session is created. Preferences are retrieved. Zone information is retrieved.

Main Success Scenario

1) User Ua enters Zone Za, which triggers Ua's Personal Agent PAa to initiate logon.

2) PAa sends logon request to Local Agent LAa with Ua's credentials.

3) LAa queries GAMS with received user credentials.

4) GAMS authenticates the user information and returns general authorisation details (Not Zone specific).

5) LAa creates a new session for Ua and returns Za information to PAa .

6) PAa sets current set of preferences, the rule base and learning strategy that are specific to Za.

7) LAa makes decisions: Include Make Decisions.

Extensions

4a. Authentication fails.

1) PA_a prompts U_a with error message.

2) U_a responds to error message.

2a. U_a enters credentials and tries again

2b. U_a aborts logon.

5a. A session already exists for U_a. LA_a returns Z_a information to PA_a.

6a. There is no existing set of preferences for Za. PAa creates an initial set of preferences for Za and set it as the current one.

6.2.3 User Account Management

Level

User-goal

Primary Actor

User

Stakeholders and Interests

- User: wants to update or provide account information.
- Global Account Management Service (GAMS): stores user account information.
- Personal Agent: provides the user an interface to GAMS.

Preconditions

The user has access to his/her personal agent.

Success Guarantee

User account is updated.

Main Success Scenario

1) User Ua enters credentials on the screen of Account management on Personal Agent PAa.

2) P Aa retrieves user account information via GAMS and displays it to Ua.

3) Ua edits account information and saves the changes.

4) PAa updates Ua's account information in GAMS.

Extensions

1a. User account does not exist yet. Ua applies for a user account and aborts account management.

6.2.4 Change of Environmental State

Level

User-goal

Primary Actor

User

Stakeholders and Interests

- User: wants to operate on building facilities to change environmental state.
- Personal Agent: to update its rule set.
- Local Agent: to provide Zone information.
- Local Authorisation Management Service (LAMS): to store local authorisation details specific to current Zone.
- Monitor&Control Agent: to enforce the operation request given by the user.

Preconditions

The user has logged on.

Success Guarantee

Request for operating a building facility succeeds. Only authorised operation is allowed and recorded as dynamic rule.

Main Success Scenario

1) User Ua requests the list of authorised operations specific to the current Zone.

2) Personal Agent PAa requests the list of authorised operations from Local Agent LAa that returns the list of available operations filtered by its LAMS.

3) Ua browses the list of operations via PAa, selects an operation, provides appropriate values for required parameters, and starts it.

4) PAa sends operation request to Monitor&Control Agent MCAa.

5) MCAa enforces the operation request: Include Set environmental state. PAa updates its rule set.

Extensions

5a. Parameter values for the requested operation are invalid.

1) MCAa returns an error message to PAa that in turn prompts Ua to check the input.

2) PAa returns to the screen of operations list.

Technology and Data Variations List

3a. Environmental parameters for temperature and humidity can be set under either actuator mode or sensor mode. Under actuator mode, the parameter value, provided by Ua, is used as the setting to actuate a building facility. Under sensor mode, the parameter value, provided by Ua, is the target environmental parameter value. MCAa needs to adjust a building facility's actuator setting to achieve this target parameter value for the space associated with Ua.

6.2.5 Preferences Modification

Level

User-goal

Primary Actor

User

Stakeholders and Interests

- User: modifies preferences.
- Personal Agent: checks preferences values against polices.

- Local Agent: provides policies to the Personal Agent. Update environmental state.

- Policy Management: stores constrains or limits to environmental parameters.

Preconditions

The user is authenticated.

Success Guarantee

Preferences are updated.

Main Success Scenario

1) User Ua browses the sets of preferences via Personal Agent PAa and provides appropriate values for updating.

2) PAa asks LAa to validate the preferences. LAa queries Policy Management to validates the preferences.

3) PAa updates the preferences.

4) LAa makes decisions: Include Make Decisions.

Extensions

2a. The preferences are invalid against certain policies.

1) PA_a prompts U_a an error message.

2) U_a adjusts preference values and tries again.

2a. U_a aborts preferences modification.

Technology and Data Variations List

*a. Preferences can have different measurements, even for the same parameter. For instance, temperature can be evaluated as predefined grades: high, comfortable and low, or as ranges such as 16 to 18 degrees.

6.3 Ontology based Agent Knowledge Structures

Multi-agent Systems are built on knowledge. From software reengineering point of view, this knowledge comes from both the domain knowledge and the software application itself.

To address these problems, some knowledge apart from the code will be required to be introduced into the software representations. For instance, the perspective of domain experts could be one of the alternative views which are deserved to be integrated into code level views.

Software comprehension refers to activities that humans perform: understanding, conceptualising and reasoning about software. The tasks of software comprehension can be classified to source code analysis tasks, document analysis tasks and source code and document trace tasks. The results of these tasks can be represented as ontologies. The obtained ontology is a useful source for software reengineering. Correction and enhancement to the original system can be done initially by modifying the ontology. Even for migrating the original system to a new environment, the obtained ontology still provides a good point to start the new design.

Ontology is a concept in philosophy. Ontology provides a shared and reusable piece of knowledge about a specific domain and has been applied in many fields, such as Semantic Web, e-commerce and information retrieval, etc. In the fields of artificial intelligence and Web, ontology is the description about domain concepts and their relations. Ontology is the theory about objects and their ties. It provides standards for differentiating kinds of objects (concrete and abstract, existent and non-existent, realistic and ideal, independent and dependent) and their ties (relations and dependency). Ontology is formal structure to support knowledge sharing and reusing. It could be used to express explicitly the semantics of structured and semi-structured information in

order to support information acquiring, maintaining and accessing automatically. Ontology provides methods to solve the heterogeneous expression of Web resource. The domain model hidden in ontology could be regarded as providing a general semantic structure for information. Since this mode of information sharing can provide a uniform communication platform as well as avoid the mistake to the same concepts with different name or title in different context, many efforts have been made to design effective tools to mediate information sharing in terms of this mode within or beyond one enterprise.

6.4 Summary

This chapter focuses on addressing the following issues, including:

- Design of a Pervasive service enabled integration architecture that allows the legacy systems to easily interact and inter-operate with other Pervasive service.
- Definition of appropriate middleware to integrate the identified components to a heavily heterogeneous Pervasive service environment.

Chapter 7
Case Study

Objectives

- To show how to use the proposed approach to develop a reengineering strategy for a building control legacy systems.
- To show how to combine Web Services and Agent techniques for software evolution in Pervasive Computing Environments.
- To show how to use related tools in software reengineering process.

7.1 Overview

The proposed agent-based service-oriented software reengineering approach includes many reverse engineering techniques and software reengineering principles. An intelligent building case study is performed to evaluate this methodology. Furthermore, the tool support for this approach and related techniques becomes an important issue.

The case study is related to decomposing legacy systems, representing as pervasive services and integrating these services into the Pervasive Computing environments. There are four core techniques proposed in this thesis for pervasive service oriented legacy software systems evolution: program transformation, program slicing, pervasive services identification and agent based service integration. In this chapter, the case study has been conducted for these proposed techniques respectively. The case study has been experimented with the proposed approach and resulting prototype.

7.2 Background

This case study is performed on a building control system. This case study aims at evaluating the proposed agent-based service-oriented software reengineering methodology, which is applied to reengineer a building control legacy system, and use the reusable pervasive service in an intelligent building control environment. Originally, it has been developed in assembly language since 1996 in Yunnan, China. Figure 7-1 shows the the archtecture description of the building control system.

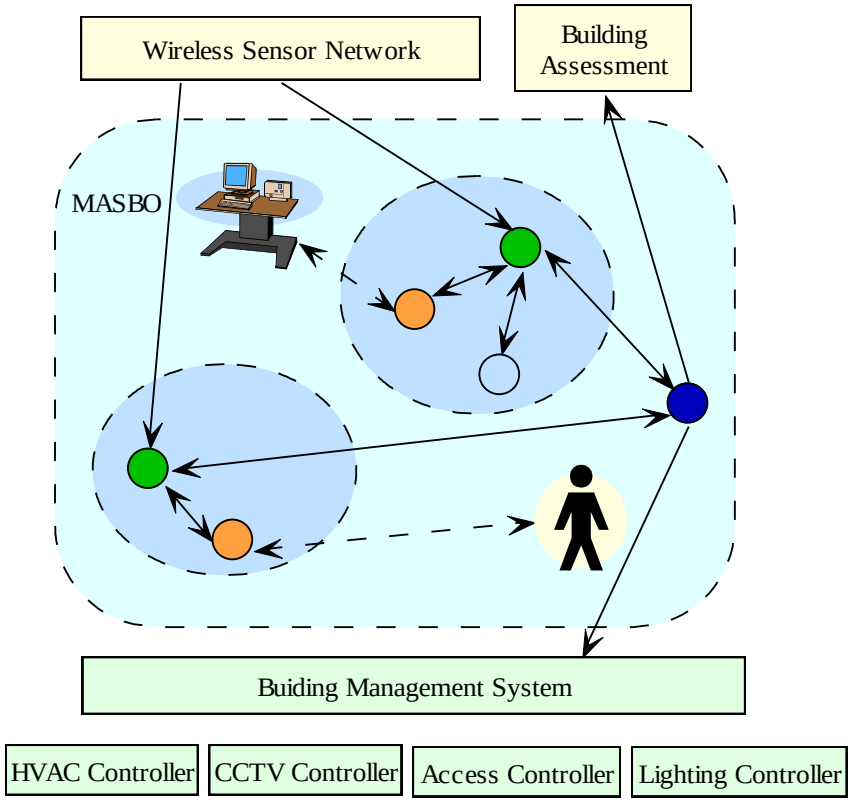


Figure 7-1. Architecture Description of Building Control System

This is an automated software system for Building Control, which can perform the following functions:

- HVAC control
- CCTV control

- Access control
- Lighting control

In this case study, the building control system is nominated as a legacy system, which intended to be reengineered and evolved into the pervasive services environment (A multi-agent intelligent building control environment), which is called MABCS (Multi-agent Building Control System). The assembly source code of this legacy building control system is shown in Appendix A. As a reengineered output, a new intelligent building control system can be established.

7.3 Service Identification

Service identification is to select related resources. Properly identifying services and determining reusable legacy components are a critical step in architecting a service oriented software reengineering task. Service identification process consists of the following two steps.

Step 1: Service identification in problem domain. It starts with a domain analysis. The goal of this domain analysis is to identify and document requirements on a set of systems in the same application domain. The domain analysis process can be carried out in two steps, sub-domain identification and analysis of the selected sub-domain. Based on the results of this domain analysis, some business functions are identified to be valuable and reusable and needed to be provided as services. In this way, some logical services are summarised, which are mainly based on requirement analysis. Direct and indirect business analysis and direct requirements analysis through stakeholder interviews and component-based modelling are an obvious and well-suited way of identifying candidate services.

Step 2: Service identification in a legacy system. There are some abstracted components in the legacy system, which encapsulate valuable functionalities and are reusable in the target service. Service-oriented component-based decomposition is archived by combining formal concept analysis and program slicing. Program

slicing are source code analysis techniques that have received great attention in the last decade for their potential applications in software engineering, especially in support of program understanding and modification. Program slicing focuses on the sub-computations performed in a program to understand the code at a lower level.

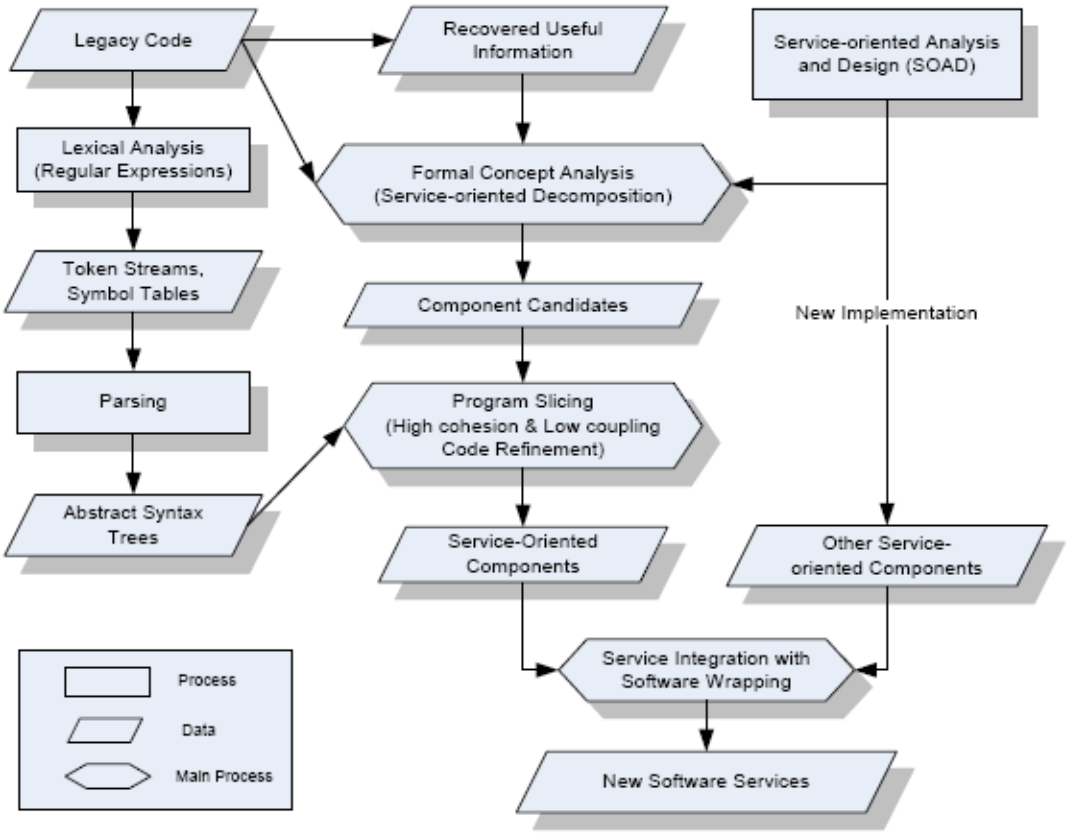


Figure 7-2.Service-oriented Component Identification

7.3.1 Assembler to WSL Translation and WSL Transformation

This case study starts with an MCS-51 assembler module that is translated to WSL and re-engineered to an abstract specification in WML. The FIP tool collects information from different sources and uses the Transformation Engine to translate the assembler files into WSL files. Once this conversion to WSL has been completed, WSL files are transformed/abstracted into WML and further extracted into UML models. These experiments have shown that programs that have been transformed using the tool can be

expressed in a higher level abstraction form that subjectively is much easier to understand than the original. Figure 7-3 shows an example of different models and views for a slice of code in assembler.

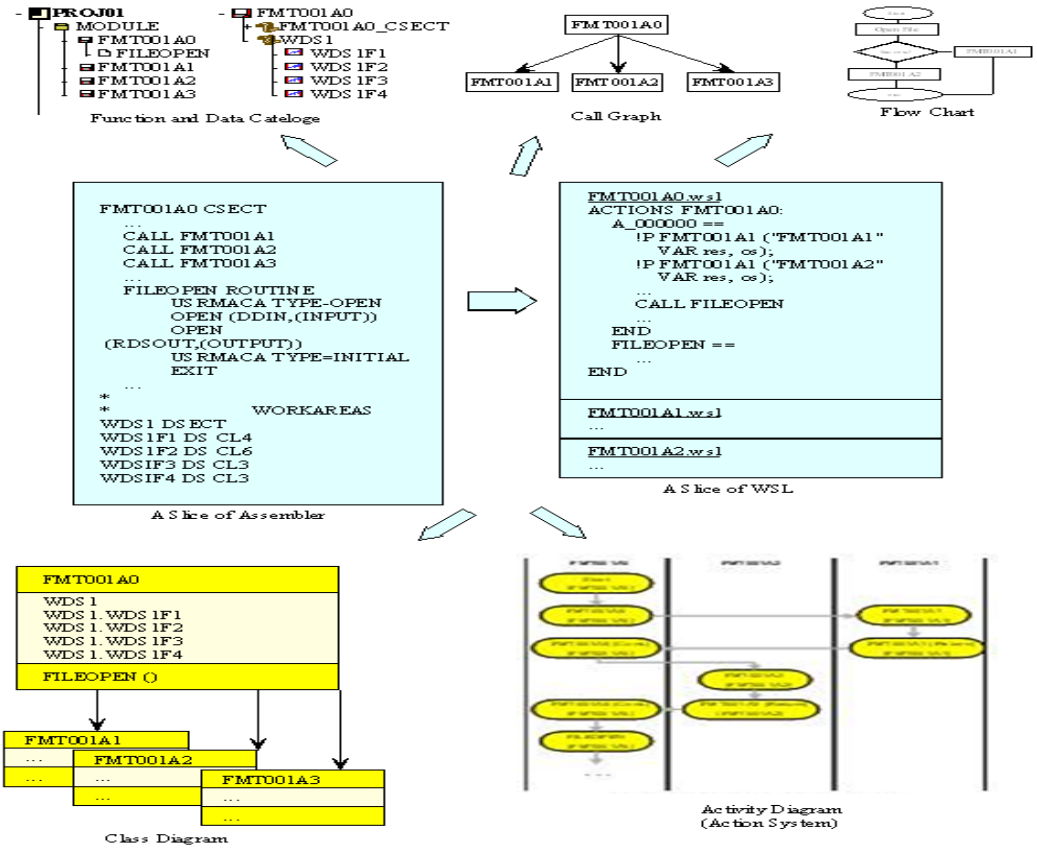


Figure 7-3. Different Models and Views for a Slice of Code

The aim of the assembler-to-WSL translator is to generate WSL code that models as accurately as possible the behaviour of the original assembler module, without worrying too much about the size, efficiency, or complexity of the resulting code. This step can be done by FermaT transformation engine. The FermaT transformation engine includes some very powerful transformations for such tasks as simplifying WSL code, removing redundancies, and tracking dispatch codes. List 7-1 shows the assembler source code file (TFM13.lst), List 7-2 shows the translated WSL source code file (TFM13.wsl) and List 7-3 shows the AST of WSL source code in XML file (TFM13.xml).

```
**********************************************************************
*   TFM13 PROGRAM - CONTROL MODULE                                   *
**********************************************************************
*  ORG  0000H
        LJMP    PSTAR
             ORG  0003H
          ; LJMP    EX0_CAN          ;;;;
        RETI
        ORG     000BH
             LJMP    P0T0

        ORG 0023H
        LJMP UARTPCR

                ORG  0030H
PSTAR:          MOV     SP,#5fh     ;59H
                MOV     IE,#1BH   ;_*0010 1011*_
                MOV     IP,#03H   ;_*0001 0000 *_
                MOV     SCON,#0
                MOV     TMOD,#21H
                MOV     TL1,#0f4H
                MOV     TH1,#0F4H
                MOV     TL0,#18H
                MOV     TH0,#0B4H
                CLR     RS0
                CLR     RS1
                MOV     R0,#7FH
                CLR     A
PSTAR1:         MOV     @R0,A
                DJNZ    R0,PSTAR1

                MOV KEYKOU,#0D0H
                 MOV     DPTR,#WKEY
                MOV A,KEYKOU
                MOVX    @DPTR,A
                MOV P1KOU,#0FFH
                MOV A,P1KOU
                MOV P1,A
                MOV LEDKOU,#0H
                MOV DPTR,#LED
                MOV A,LEDKOU
                MOVX    @DPTR,A
                MOV DPTR,#PC00+1
                MOVX    A,@DPTR
                JNB   ACC.7,PSTAR00
                MOV A,P1KOU
                CLR   ACC.5
                MOV P1KOU,A
                CLR   P1.5
                PSTAR00:        MOV     DPTR,#0
PSTAR0:          CLR A
        MOVX    @DPTR,A
                INC   DPTR
                MOV A,DPH
                CJNE A,#10H,PSTAR0
                MOV DPTR,#2000H
PSTAR2:          MOV     A,#0FFH
        MOVX    @DPTR,A
                INC   DPTR
                MOV A,DPH
                CJNE A,#50H,PSTAR2

                MOV DPTR,#HJWRH
                MOV A,#20H
                MOVX    @DPTR,A
                MOV DPTR,#HJRDH
                MOV A,#20H
                MOVX    @DPTR,A
                MOV DPTR,#GZWRH
                MOV A,#20H;???????????????
```

```
                MOVX    @DPTR,A
                MOVX    @DPTR,A
                MOV DPTR,#GZRDH
                MOV A,#20H;??????????????????????//
                MOVX    @DPTR,A
                MOV DPTR,#LDWRH
                MOV A,#40H
                MOVX    @DPTR,A
                MOV DPTR,#LDRDH
                MOV A,#40H
                MOVX    @DPTR,A
                MOV DPTR,#CANFRDH
                MOV A,#50H
                MOVX    @DPTR,A
                MOV DPTR,#CANFWRH
                MOV A,#50H
                MOVX    @DPTR,A
                MOV DPTR,#CANSRDH
                MOV A,#58H
                MOVX    @DPTR,A
                MOV DPTR,#CANSWRH
                MOV A,#58H
                MOVX    @DPTR,A
                MOV DPTR,#T232RDH
                MOV A,#60H
                MOVX    @DPTR,A
                MOV DPTR,#T232WRH
                MOV A,#60H
                MOVX    @DPTR,A
                MOV DPTR,#R232RDH
                MOV A,#70H
                MOVX    @DPTR,A
                MOV DPTR,#R232WRH
                MOV A,#70H
                MOVX    @DPTR,A
                SETB DSET
                MOV MENUX,#0
                MOV     COLORB,#00H     ;
                MOV CDWH,#0
                MOV CDWL,#0
                MOV R4,#0A0H
        LCALL   CLEAR
        CPL WDOG
        ;MOVDPTR,#WKEY
        ;MOVX    A,keyout;@DPTR
        ;SETB    ACC.4
        ;MOVX    @DPTR,A
        MOV DPTR,#8800H
        MOVX    A,@DPTR
        MOV PNUM,A
                LCALL   HM01
                LCALL TONGXUN_INI_PC
                ;LCALL  HM02
                LCALL   PTIME1
                CLR BA16
                LCALL   LOCKC
                MOV     BGBUF,#180
                CPL WDOG
                SETB TR0
        SETB EA
        MOV R3,#250
           MOV  R2,#0
PSTAR3:         JB  MSET,PSTAR44
        DJNZ R2,PSTAR3      ;AJMP   PSTAR3;

PSTAR66: SETB DSET
        DJNZ R3,PSTAR3
        ;MOVCDWH,#0F0H
        ;MOVCDWL,#0C8H
        ;MOVDPTR,#ZJDA
        ;MOVZCON,#12
```

```
            ;LCALL      PQWM
            ;LCALL      DELAY15
            AJMP PSTAR7

PSTAR44:    CLR  DSET
            MOV DPTR,#MSTATE
            MOVX        A,@DPTR
            CJNE A,#55H,PSTAR66
            ;CLR  A
            ;MOVX       @DPTR,A
            SETB DSET

PSTAR8:             JB      MSET,PSTAR4
            AJMP PSTAR8

PSTAR6:             JZ      PSTAR7
            CLR  ACC.7
            PUSH ACC
            CLR  A
            MOVX        @DPTR,A
            SETB DSET
            MOV CDWH,#0E8H
            MOV CDWL,#0C8H
            MOV DPTR,#ZJDA1
            MOV ZCON,#10
            LCALL       PQWM
            DEC  CDWH
            MOV CDWL,#0E0H
            POP  ACC
            LCALL       PW2D
            LJMP PSTAR8
PSTAR9:             SETB DSET
            LJMP PSTAR8

PSTAR4:              CLR DSET
            CPL  WDOG
            MOV DPTR,#MANSWER
            MOVX        A,@DPTR
PSTAR5:             CJNE A,#55H,PSTAR6
            CLR  A
            MOVX        @DPTR,A
PSTAR7:             SETB DSET

            LCALL       HM02
MAIN:
            JNB  BGJSJ,MAIN0
            JNB  BGJTT,MAIN0
            CLR  BGJTT
            LCALL       PGJSJ
            AJMP MAIN1


MAIN0:              ;JB     BEVENT,MAIN1
            JB      BPRGM,MAIN3

                    ;MOV         A,MBUF
                    PUSH        CDWH
                    PUSH        CDWL
                    SETB BA16
                    MOV COLORB,#0E2H
                    MOV         CDWH,#0ACH
                    MOV         CDWL,#4FH
                    JNB  MSET,$
                    CLR  DSET
            MOV DPTR,#SKTIME+5
            MOVX        A,@DPTR
                    XRL  A,MBBUF
                    JZ      MAIN00
                    LCALL      PTIME1;     --S
MAIN00:              SETB        DSET
            CLR  BA16
```

```
                POP   CDWL
                POP   CDWH
MAIN3:           JNB        PBG,MAIN1
                CLR       PBG
                MOV       A,BGBUF
                JZ        MAIN1
                DEC       BGBUF
                DEC  A
                JNZ   MAIN1
                MOV A,KEYKOU
          CLR   ACC.6
          MOV KEYKOU,A
          MOV DPTR,#WKEY
          MOVX      @DPTR,A
          ;JNB  BPRGM,MAIN1
          MOV A,HMTU
          CJNE A,#27,MAIN4
          CLR   BGJSJ
          MOV A,#CEXIT
          LCALL     PSAVE

          MOV MENUX,#5
MAIN4:          LCALL     PRESTOR

MAIN1:          JNB   BKEY02,MAIN2
          LCALL     KEYOUT
          CPL   WDOG
          CLR   BKEY02
MAIN2:          LCALL     SKCHECK
          CPL WDOG
          LCALL MAIN_PC
          MOV A,PNUM
          ORL   A,#01H
          JNZ   MAIN5
          ;LCALL     RECAN
MAIN5:          ;LCALL     CANSCAN
          LCALL     SEND232

          NOP
          CPL   WDOG
AA:             LJMP      MAIN
send232:        ret
;.....................
RECAN:          MOV DPTR,#CANSRDL
          MOVX      A,@DPTR
          MOV B,A
          DEC   DPL
          DEC   DPL
          MOVX      A,@DPTR
          INC   DPL
          CJNE A,B,RECAN01A

          MOVX      A,@DPTR
          MOV B,A
          DEC   DPL
          DEC   DPL
          MOVX      A,@DPTR
          ADD  A,#58H
          CJNE A,B,RECAN01B
          RET
RECAN01A:
          MOVX      A,@DPTR
          MOV R7,A
          MOV A,B
          MOV R6,A
          AJMP RECAN02
RECAN01B:       MOV DPTR,#CANSRDL
          MOVX      A,@DPTR
          MOV R6,A
          MOV A,B
          MOV R7,A
```

```
RECAN02:  JNB   MSET,$
                CLR   DSET
                MOV DPH,R7
                MOV DPL,R6
                MOVX        A,@DPTR
                CJNE A,#01H,RECAN03
                MOV DPTR,#SHJNUMH+1
                AJMP RECAN06A
RECAN03:          CJNE      A,#02H,RECAN04
                MOV DPTR,#SLDNUMH+1
                AJMP RECAN04A
RECAN04:            CJNE      A,#03H,RECAN05
RECAN04A:           MOV       DPTR,#SHSNUMH+1
                MOVX        A,@DPTR
                INC   A
                MOVX        @DPTR,A
                MOV DPTR,#SDACT
                AJMP RECAN07
RECAN05:          CJNE      A,#06H,RECAN06
                MOV DPTR,#SHJNUMH+1
                AJMP RECAN06A
RECAN06:            MOV       DPTR,#SGZNUMH+1
RECAN06A:           MOVX    A,@DPTR
                INC   A
                MOVX        @DPTR,A
                MOV DPTR,#SDFIRE
RECAN07:  MOV R5,DPH
          MOV R4,DPL
          LCALL       GZMOVE0
                CLR   DSET
                MOV DPTR,#CANSRDL
                MOVX        A,@DPTR
                ADD  A,#20H

                MOVX        @DPTR,A
                DEC  DPL
                MOVX        A,@DPTR
                ADDC        A,#0
                MOVX        @DPTR,A
                LCALL       REVENT
                LCALL       STOR66
                RET
```

List 7-1. Assembler Source Code of TFM13

```
VAR < cc := 0, cc1 := 0, destination := 0 >:
ACTIONS _enter_:
   _enter_ ==
      C:" <ENTRY POINT> ";
      C:" <NAME=TFM13> ";
      r7 := __r7_init__;
      r11 := __r11_init__;
      r12 := __r12_init__;
      r13 := __r13_init__;
      r14 := __r14_init__;
      CALL TFM13 END
   _start_ == CALL Z END
   TFM13 ==
      C:"*************************************************************************";
      C:"*TFM13 PROGRAM - CONTROL MODULE                                        *";
      C:"*************************************************************************";
      C:"*";
      C:"*";
```

```
    C:"*              PRINT NOGEN";
    CALL A_000000 END
A_000000 ==
    C:"<FermaT> 00000035 ";
    !P push_regs(r0,
                 r1,
                 r2,
                 r3,
                 r4,
                 r5,
                 r6,
                 r7,
                 r8,
                 r9,
                 r10,
                 r11,
                 r12,
                 r13,
                 r14
       VAR reg_stack);
    CALL A_000004 END
A_000004 ==
    C:"<FermaT> 00000036 "; r3 := r15; CALL A_000006
END
A_000006 ==
    C:"<FermaT> 00000038 WSAVE WSAVE[5] WSAVE[6] WSAVE[7] WSAVE[8]";
    WSAVE[5..8] := r13;
    CALL A_00000A END
A_00000A ==
    C:"<FermaT> 00000039 WSAVE";
    r14 := !XF address_of(WSAVE);
    CALL A_00000E END
A_00000E ==
    C:"<FermaT> 00000040 ";
    a[r13 + 8, 4] := r14;
    CALL A_000012 END
A_000012 ==
    C:"<FermaT> 00000041 WSAVE";
    r13 := !XF address_of(WSAVE);
    CALL A_000016 END
A_000016 ==
    C:"*";
    C:"<FermaT> 00000043 WTOTAL";
    !P zap(!XF p_lit(1, 1, "0") VAR WTOTAL);
    IF !XC dec_eq(WTOTAL, !XF p_lit(1, 4, "0"))
       THEN cc := 0
    ELSIF !XC dec_less(WTOTAL, !XF p_lit(1, 4, "0"))
       THEN cc := 1
       ELSE cc := 2 FI;
    CALL A_00001C END
A_00001C ==
    C:"<FermaT> 00000044 WCT";
    !P ap(!XF p_lit(1, 1, "1") VAR WCT);
    IF !XC dec_eq(WCT, !XF p_lit(1, 4, "0"))
       THEN cc := 0
    ELSIF !XC dec_less(WCT, !XF p_lit(1, 4, "0"))
       THEN cc := 1
       ELSE cc := 2 FI;
```

```
    CALL A_000022 END
A_000022 ==
    C:"<FermaT> 00000045 WNUM";
    !P ap(!XF p_lit(1, 1, "1") VAR WNUM);
    IF !XC dec_eq(WNUM, !XF p_lit(1, 4, "0"))
       THEN cc := 0
    ELSIF !XC dec_less(WNUM, !XF p_lit(1, 4, "0"))
       THEN cc := 1
       ELSE cc := 2 FI;
    CALL A_000028 END
A_000028 ==
    C:"*";
    C:"<FermaT> 00000047 ";
    !P FMT001A1("FMT001A1" VAR result_code, os);
    r15 := result_code;
    CALL A_000036 END
A_000036 ==
    C:"<FermaT> 00000054 ";
    !P FMT001A2("FMT001A2" VAR result_code, os);
    r15 := result_code;
    CALL A_000046 END
A_000046 ==
    C:"<FermaT> 00000061 ";
    !P FMT001A3("FMT001A3" VAR result_code, os);
    r15 := result_code;
    CALL A_000056 END
A_000056 ==
    C:"*";
    C:"<FermaT> 00000072 WSAVE WSAVE[5] WSAVE[6] WSAVE[7] WSAVE[8]";
    r13 := WSAVE[5..8];
    CALL A_00005A END
A_00005A ==
    C:"<FermaT> 00000073 ";
    !P pop_regs(
       VAR r0, r1,
          r2,
          r3,
          r4,
          r5,
          r6,
          r7,
          r8,
          r9,
          r10,
          r11,
          r12,
          r13,
          r14,
          reg_stack);
    CALL A_00005E END
A_00005E ==
    C:"<FermaT> 00000074 ";
    r15 := 0;
    IF r15 = 0 THEN cc := 0 ELSE cc := 2 FI;
    CALL A_000060 END
A_000060 ==
    C:"<FermaT> 00000075 ";
    destination := r14;
```

```
      CALL dispatch END
   dispatch ==
      IF destination = 0
         THEN CALL Z
         ELSE C:" Unknown destination "; CALL Z FI END
ENDACTIONS ENDVAR
```

List 7-2. WSL Source Code of TFM13

```
<Statements>
    <Var>
        <Assigns>
            <Assign>
                <Var_Lvalue value="cc"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
            <Assign>
                <Var_Lvalue value="cc1"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
            <Assign>
                <Var_Lvalue value="destination"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
        </Assigns>
        <Statements>
            <A_S>
                <Name value="_enter_"></Name>
                <Actions>
                    <Action>
                        <Name value="_enter_"></Name>
                        <Statements>
                            <Comment value="&lt;ENTRY POINT&gt; "></Comment>
                            <Comment value="&lt;NAME=FMT001A0&gt; "></Comment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r7"></Var_Lvalue>
                                    <Variable value="__r7_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r11"></Var_Lvalue>
                                    <Variable value="__r11_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r12"></Var_Lvalue>
                                    <Variable value="__r12_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r13"></Var_Lvalue>
                                    <Variable value="__r13_init__"></Variable>
                                </Assign>
```

```
                        </Assignment>
                        <Assignment>
                            <Assign>
                                <Var_Lvalue value="r14"></Var_Lvalue>
                                <Variable value="__r14_init__"></Variable>
                            </Assign>
                        </Assignment>
                        <Call value="FMT001A0"></Call>
                    </Statements>
                </Action>
                <Action>
                    <Name value="_start_"></Name>
                    <Statements>
                        <Call value="Z"></Call>
                    </Statements>
                </Action>
                <Action>
                    <Name value="FMT001A0"></Name>
                    <Statements>
                        <Comment
                                </Statements>
                            </Guarded>
                        </Cond>
                    </Statements>
                </Action>
            </Actions>
        </A_S>
    </Statements>
</Var>
</Statements>
```

List 7-3. TFM13 AST in XML

7.3.2 Program Slicing Analysis

The building control system is composed of a set of programs related through external calls and it can be represented by asset of modules and a set of call relationships between modules, where each module is represented by a circle and each call relationship is represented by a line connecting two modules. The graph nodes represent the programs and it sedges depict the call relation between programs. This representation is referred as system call graph.

Program or subsystems in this building control system could be structured in a set of subroutines related through internal calls. The call relation on the subroutines of a program can be represented by a graph whose nodes correspond to the program subroutines and edges depict the internal calls. This representation is referred as a

program call graph. This call graph architecture describes abstracted components in the legacy system, which encapsulates valuable functionalities and is reusable in the target application or service, as well as their relationship. Figure 7-4 gives the system call graph of building control system.

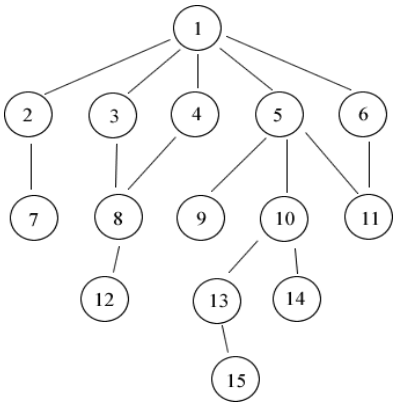


Figure 7-4. System Call Graph of Building Control System

The original program fragment of user login subroutine is shown as List 7-4.

```
;*******************************************************************
;*   TFM13 PROGRAM – USER LOGIN MODULE                             *
;*******************************************************************
;*****************************************************************
HJ4000: MOV A,NETNO              ;(M12C)
        JNZ HJ4100               ;MACHINE NUMBER
        MOV DPTR,#46E0H          ;NETFIRE WRITE POINT(LO)
        MOVX A,@DPTR
        MOV R7,A
        INC DPL                  ;NETFIRE READ POINT(LO)
        MOVX A,@DPTR
        MOV R1,A
HJ4001: CJNE A,07H,HJ4002        ;NO NETFIRE TO HJ1000
        LJMP HJ1000
HJ4002: MOV P2,#6BH
        MOVX A,@R1
        MOV 42H,A
        INC R1
        MOVX A,@R1
        MOV 43H,A
        INC R1
        MOVX A,@R1
        MOV 45H,A
        INC R1
        MOVX A,@R1
        MOV 49H,A
        INC R1
        MOVX A,@R1
        MOV 44H,A
        INC R1
        CJNE R1,#8CH,HJ4004
        MOV R1,#00H
HJ4004: LCALL HJCZ1
        NOP
        MOV A,R1
```

```
            CJNE A,07H,HJ4002
            MOV DPTR,#46E1H
            MOVX @DPTR,A
            LJMP HJ1000
HJ4100: MOV 4DH,A                   ;(M12D)
            MOV DPTR,#46E4H          ;NETCONTROL WRITE POINT(LO)
            MOVX A,@DPTR
            MOV R7,A
            INC DPL                  ;NETCONTROL READ POINT(LO)
            MOVX A,@DPTR
            MOV R1,A
HJ4104: CJNE A,07H,HJ4102        ;NO NETCONTROL TO HJ1000
            LJMP HJ1000
HJ4102: MOV P2,#6BH
            MOVX A,@R1
            MOV 41H,A                ;(M12A)
            INC R1
            MOVX A,@R1
            MOV 46H,A
            INC R1
            MOVX A,@R1
            MOV 47H,A
            INC R1
            MOVX A,@R1
            MOV 48H,A
            INC R1
            CJNE R1,#40H,HJ4106
            MOV R1,#00H
HJ4106: LCALL HJCZ3
            NOP
            MOV A,R1
            CJNE A,07H,HJ4102
            MOV DPTR,#46E5H
            MOVX @DPTR,A
            LJMP HJ1000
;*************************************************************
```

List 7-4. Original Program Fragment of User Login Subroutine

The Figure 7-5 presents the control flow graph of user login subroutine.

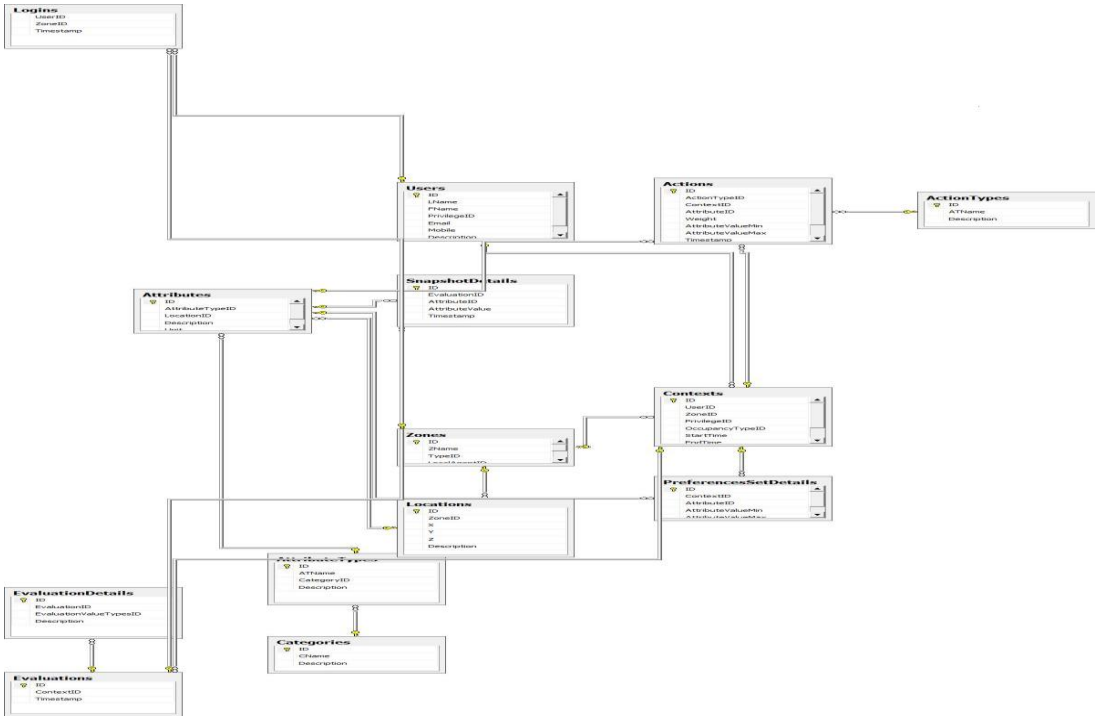


Figure 7-5. Control Flow Graph of User Login Subroutine

List 7-5 shows the slice of the user login subroutine.

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
      CodeFile="NewUserPreferences.aspx.cs" Inherits="UserManagement_NewUserPreferences"
      Title="Create preferences" EnableEventValidation="false" %>

<%@ MasterType VirtualPath="~/MasterPage.master" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContentPlaceHolder" runat="Server">
      <table style="width: 600px; height: 320px">
            <tr>
                  <td style="width: 50px">
                  </td>
                  <td style="width: 490px">
                  </td>
                  <td>
                  </td>
            </tr>
            <tr>
                  …
            <tr>
                  <td style="width: 50px">
                  </td>
                  <td style="width: 490px">
                  </td>
                  <td>
                  </td>
            </tr>
      </table>
</asp:Content>
      using System;
      using System.Data;
      using System.Configuration;
      using System.Collections;
      using System.Web;
      using System.Web.Security;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class UserManagement_ListUserContexts : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            Master.HeadingLabelText += " - List user contexts";
        }

    }

    protected void ContextsGridView_RowUpdated(object sender, GridViewUpdatedEventArgs e)
    {
        if (e.Exception != null)
        {
            Master.MessageLabelText = "Cannot update record";
            e.ExceptionHandled = true;
        }
        else
        {
            Master.MessageLabelText = "Record updated";
        }
    }
    protected void ContextsGridView_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
    {
        Master.MessageLabelText = "Edit canceled";
    }
    protected void ContextsGridView_RowDeleted(object sender, GridViewDeletedEventArgs e)
    {
        if (e.Exception != null)
        {
            Master.MessageLabelText = "Cannot delete record";
            e.ExceptionHandled = true;
        }
        else
        {
            Master.MessageLabelText = "Record deleted";
        }
    }
    protected void ContextsGridView_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if (e.CommandName == "Edit")
        {
            Master.MessageLabelText = "";
        }
    }

    protected void ContextsGridView_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            string value = DataBinder.Eval(e.Row.DataItem, "ID").ToString();

            ObjectDataSource s = (ObjectDataSource)e.Row.FindControl("ObjectDataSource8");
            s.SelectParameters[1].DefaultValue = value;

        }
    }
}
```

List 7-5. Slice of User Login Subroutine

Based on the slicing approach, the legacy building control system programs is deeply

understand and they have been decomposed into several concerned legacy code segments. The dead codes are eliminated from the legacy system and the selected code segments function is independent.

7.3.3 Clustering Analysis

In order to group the sliced legacy code segments and create a hierarchical structure of them, the cluster analysis is applied. The clustering analysis is carried out to identify legacy functionalities, such as project creation, discussion forum, e-publishing and so on. Architects supervise the clustering analysis process and select the cutting point.

Figure 7-6 shows its class diagram. This clustering method together with human supervision provides a powerful analysis on legacy systems, and represents them into modules.

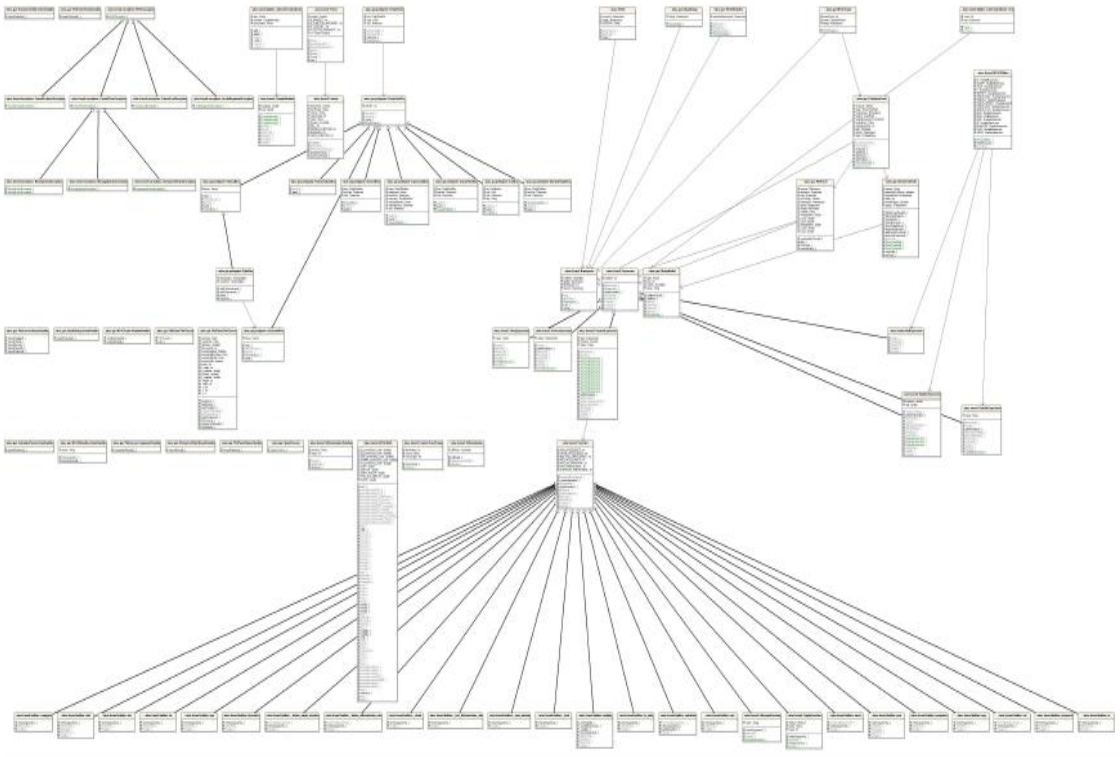


Figure 7-6. The Class Diagram of Building Control System

7.3.4 Wrapping as Pervasive Service

Some extracted legacy components can be wrapped as Web services straightforward according to requirements. The goal of the wrapping process is to provide the component extracted from the legacy code with a WSDL interface. The technique used is to transform each entry into a method and to transform each parameter into an XML data element. The data structures will become complex elements with one or more sub-elements. The methods will have their arguments and results as references to the data element descriptions. Both the methods and the parameters will be built into an XML schema with a SOAP framework.

```
<!—This schema was generated from prog:inputs\xm059i.cpy
by the Prototype Tool SOWrap on date:030206—>

<schema name = "xm059i" xmlns= "XSDCOB">
    <XSDCOB:complexType type = "#file" name = "xm059i"
                        content = "eltOnly" model = "closed">
        <XSDCOB:complexType type – "#params" name – "XM059–PARAMS"
            content = "eltOnly" model = "closed" level = "01"
        occurs = "ONEORMORE" minOccurs = "1" maxOccurs = "unbounded">

        <XSDCOB:element type = "#dec" name = "P1–TT"
            content = "TextOnly" model = "closed" level = "03"
            occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
            pos = "0001" lng = "0002" pic = "99" usage = "DISPLAY"/>

        <XSDCOB:element type = "#dec" name = "P1–MM"
            content = "TextOnly" model = "closed" level = "03"
            occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
            pos = "0003" lng = "0002" pic = "99" usage = "DISPLAY"/>

        <XSDCOB:element type = "#dec" name = "P1–JJ"
            content = "TextOnly" model = "closed" level = "03"
            occurs = "ONEORMORE" minOccurs = "0001" maxOccurs = "0001"
            pos = "0007" lng = "0002" pic = "99" usage = "DISPLAY"/>

        </XSDCOB:complexType>
    </XSDCOB:complexType>
</schema>
```

List 7-6. XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
....
<training>
    <title>Time Management</title>
    <time>9:05</time>
    <duration>3 hours</duration>
    <number>000042</number>
    <location>Gateway House 5.31</location>
    <parking>Main car park</parking>
</training>
<training>
    <title>First Aid Person</title>
    <time>13:10</time>
    <duration>4 hours</duration>
    <number>000050</number>
    <location>Hawthorn Building 1.52</location>
    <parking>Small car park</parking>
</training>
```

List 7-7. XML Fragment for Training Example

7.4 Agent-based Service Integration

What is interesting and inspiring in this research is the application of MAS to balance energy saving and occupants' preferences, which in our view is one of most important features for an intelligent building environment that emphasises both energy sustainability and human wellbeing. On the other hand, while the presented system is capable of adjusting the heating and light level to meet personal preferences, these preferences are predefined and cannot be adapted or learned according to the feedback or behaviour of the occupants. It can detect a person's presence and adapt the room environment settings according to his/her preferences via an active badge system. The badge system itself, however, does not provide the means to distinguish between actuations from different occupants, which are necessary for occupants' behaviour learning mechanisms proposed in other's work.

Besides the learning ability, this research also presents another feature in some cases preferable for intelligent building environment: user interaction with and feedback to

the MAS.

The MAS asks for occupant's preference for any given programmable setting and tries to adjust its rule set to achieve this setting. The occupant is then asked to confirm the environment change, the result of which will help the MAS to either re-adjust or accept the current rule set.

While this research presents two important features as mentioned earlier, its use of embedded agents makes it difficult to take advantage of sophisticated agent platforms and as claimed by the researchers, places severe constraints on the possible AI solutions.

Following the work in [113] is the iDorm project [49], where an intelligent dormitory is developed as a test bed for a multiuse ubiquitous computing environment. One of improvements of iDorm over is the introduction of iDom gateway server that allows a standard interface to all the room's sub networks by exchanging XML-formatted queries with all the principal computing components, overcoming many of the practical problems of mixing networks. This improvement presents another feature for applying agent technology in intelligent buildings, i.e., combining IT network with building control network to maximally utilise the sensory data and control ability.

However, iDom is still based on embedded agents as those in [113], which though demonstrating learning and autonomous behaviours, are running on nodes with very limited capacity compared to a PC or workstation. No agent platforms or advanced techniques can be used.

In addition, the requirement of user feedback or interactions in intelligent building environment is controversial. Some researchers claim that ambient intelligence should not be intrusive, i.e., no special devices used and no imposing rules on occupants' behaviour. In [112], a multi-agent system is presented for intelligent building control. In contrast to the approach in [113] [49], the MAS is equipped with an unsupervised online real-time learning algorithm that constructs a fuzzy rule-base, derived from very sparse data in a non-stationary environment.

Learning algorithm in this research is completely unsupervised. All feedback is acquired by means of observing occupants' behaviours without intruding them. While avoiding intrusiveness could be preferable in some cases, the MAS loses the ability to distinguish between actuations and preferences from different occupants, and thus the preferences learned are not coupled with the occupants but the room they are in. By this way, it is unable to take into account personal preferences.

In contract to the approach in [113] [49], the agents are implemented using advanced agent techniques and reside on PC servers. All agents run within an agent building and learning environment (ABLE) framework, which implements the Java agent service (JAS) specification that implements the foundation of intelligent physical agents (FIPA) abstract agent architecture.

7.4.1 Service Integration Process

A software service is a computational or data serving process, which has a well-defined functional interface and can be easily discovered and accessed. It processes certain well-defined XML documents what it receives through some combination of transport and application protocols [151]. Service integration is to deploy legacy components and other newly-built components in a service provider agent. It is a necessary step to bridge legacy service candidates into a functional software service, because service-oriented architectures encourage individual services to be self-contained. Normally a service integration process includes the following steps.

- Step 1: Legacy code refinement. According to the recovered architecture information, reusable component and connectors are extracted for service integration. Although the legacy code is independent and loose coupled, it may still has some coupled relationship with other software entities. This refinement eliminates unrelated interaction interfaces and specifies the interface details for service internal communication. In this refinement, dead code is detected and removed.

- Step 2: New functional component development. New business requirements

found in service candidate identification lead to construct some new components. These new components are designed according to the difference between the domain model obtained from domain analysis and the architecture recovered from the legacy code. They will be integrated in the target service to extend the business logic.

• Step 3: Connector development. These connectors are designed to connect the fashioned legacy components and newly-built functional components. They intercept ingoing and outgoing calls and replace them by appropriate interaction. These connectors are mainly glue code and wrappers for service integration.

• Step 4: Service interface and description development. A service interface is the abstract boundary that a service exposes. It defines the types of messages and the message exchange patterns that are involved in interacting with the service, together with any conditions implied by those messages. Furthermore, a service must have sufficient service descriptions, which are associated with the service to enable its use by other parties.

Ideally, a service description will give sufficient information so that an automatic agent may not only use the service but also decide if the service is appropriate; that in turn implies a description of the semantics of the service. Because Web services are network accessible interfaces to application functionality, built using standard Internet technologies, a Web service interface is applicable to most software services. If the target service is to provide as a Web service, a rigorous definition of the functionality of the service is provided in Web Services Description Language (WSDL). WSDL also describes the operations that a service can support and the parameters each operation accepts and returns.

• Step 5: Transport mechanism integration. Because a service is message oriented, the transport mechanism is necessary for a service. To implement a Web service, a SOAP processor will be integrated. It supports message transfer on the Web in a firewall friendly way.

• Step 6: Service complexity reduction. This reduction focuses on minimising

inter-component communication within the service and improving the systems maintainability and quality of the service. Software metrics and dependence analysis techniques are used in this step. In order to describe this service integration process in more details, the wrapping techniques are divided into two categories. One is to wrap legacy components as Web services straightforward; the other is to integrate legacy components with other newly-built components in service providers.

7.4.2 Multi-Agent Based Service Integration

There is no formal definition of an multi-agent system, only an agreement on the most common features like multiple agents acting in one environment, all agents have the same input, actions affect the common environment of all agents or communication between agents and probably between agents and the environment.

On the other hand are there some definitions of a multi-agent system which are rather descriptive like: An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. Multi-agent systems can be claimed to include human agents as well. Human organisations and society in general can be considered an example of a multi-agent system.

Multi-agent systems can manifest self-organisation and complex behaviours even when the individual strategies of all their agents are simple.

7.4.2.1 Agent-based Service Integrator Specification

The legacy components are not always wrapped as Web services straightforward, so they are integrated in the agent-based integrator with other newly-built services, which are to satisfy new business requirements according to the application domain model.

List 7-8 Show the Agent-based service integrator specification.

```
/*
 * Main.java
  * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */
```

```
package integrator;

import com.thalesresearch.cmips.messagebus.*;
import com.thalesresearch.cmips.wsnmessages.*;
import integrator.domain.*;
import integrator.persistence.WsnQueryMessageSqlMap;
import java.io.*;
import java.util.ArrayList;
import static integrator.domain.WsnQueryMessageBean.*;

public class Integrator implements MessageBusSocketCallback {

	private static MessageBusSocket ccmb;

		/* basic initialisation stuff. Create a socket on the message bus and
		 * connect to it (server or client as specified)
		 */
	public void connect(String url) {
		this.ccmb = new MessageBusSocket(this);
		if (!ccmb.connectAsClient(url == null ? "127.0.0.1" : url)) {
			System.err.println("Client connect() failed.");
			System.exit(1);
		}
	}

	public void receiveMessage(Object obj, int connId) {
		System.out.println("Receive Message: " + ((WsnDataMsgLoneDouble) obj).getParametersAsString());
		importWsnDataMessageSingleBean((WsnDataMsgLoneDouble) obj);
	}

	public void connectionClosedNotification(int connectionId) {

	}

		/*
		 * Main class to make the test runnable
		 * @param args: need true for client side test app, false for gateway side
		 */
	public static void main(String[] args) {

		boolean done;

		Integrator integrator = new Integrator();

		ArrayList<WsnQueryMessageBean> wqmbl = integrator.getWsnQueryMessageBeanList();

		done = false;
		BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
		String str;

//		prepareWsnQueryMessagePeriodicBean();

		integrator.connect(args.length == 1 ? args[0] : null);

		// read fom command line and send through message bus
		int count = 0;
		while(!done) {
			try {
/*				str = in.readLine();
				if (str == null) {
					System.out.println("null read from console.");
					break;
				}
```

```
 */
                        if ((count = integrator.executeWsnQueryMessages()) != -1)
                            System.out.format("-> completed sending: %d messages\n", count);
                        else
                            System.err.println("-> sending failed");
                        // sleep for 1000 milliseconds
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                    } //catch (IOException e) {}
                }
                try {
                    in.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
        }
        // only for producing dummy queries
        static private WsnQueryMessagePeriodicBean prepareWsnQueryMessagePeriodicBean() {

                WsnQueryMessagePeriodicBean wqmpb = new WsnQueryMessagePeriodicBean();

                wqmpb.setSensingMode(WsnServiceOntology.SENSING_MODE_TEMPERATURE);
                wqmpb.setSamplingInterval(2);
                wqmpb.setZoneId(3);
                wqmpb.setLocationHeight(WsnServiceOntology.LOCATION_HEIGHT_FLOOR);
                wqmpb.setLocationSemantic(WsnServiceOntology.LOCATION_SEMANTIC_NEARWINDOW);
                wqmpb.setSpatialAggregation(WsnServiceOntology.SPATIAL_AGGREGATION_AVERAGE);
                wqmpb.setReportInterval(20);
                wqmpb.setDataProcessing(WsnServiceOntology.DATA_PROC_AVERAGE);

                WsnQueryMessageSqlMap.insertWsnQueryMessage(wqmpb);
                WsnQueryMessageSqlMap.insertWsnQueryMessagePeriodic(wqmpb);

                return wqmpb;
        }

        /* Import data message into integrator database
         * @param wdms: WSN data message
         */
        private void importWsnDataMessageSingleBean(WsnDataMsgLoneDouble wdms) {
                WsnDataMessageSingleBean wdmsb = new WsnDataMessageSingleBean();
                wdmsb.setDataValue(wdms.getDataValue());
/*              wdmsb.setLocationX(wdms.getLocationX());
                wdmsb.setLocationY(wdms.getLocationY());
                wdmsb.setLocationZ(wdms.getLocationZ());*/
                wdmsb.setSubZoneId(wdms.getSubZoneId());
                wdmsb.setTimestamp(wdms.getTimestamp());
                wdmsb.setQueryId(wdms.getQueryId());
                wdmsb.setSensingMode(wdms.getSensingMode());
                wdmsb.setZoneId(wdms.getZoneId());

                WsnQueryMessageSqlMap.insertWsnDataMessageSingle(wdmsb);
        }

        private ArrayList<WsnQueryMessageBean> getWsnQueryMessageBeanList() {
                return WsnQueryMessageSqlMap.selectWsnQueryMessageList(false);
        }

        private void executeWsnQueryMessage(int queryId) {
                WsnQueryMessageSqlMap.executeWsnQueryMessage(queryId);
        }

        private int executeWsnQueryMessages() {
                WsnQueryMessagePeriodic wqmp;
```

```
        WsnQueryMessagePeriodicBean wqmpb;

        WsnQueryMessageGetAll wqmga;

        WsnQueryMessageCancel wqmc;
        WsnQueryMessageCancelBean wqmcb;

        WsnQueryMessageAlarm wqma;
        WsnQueryMessageAlarmBean wqmab;

//          WsnQueryMessagePeriodicBean wqmpb;
//          wqmpb = prepareWsnQueryMessagePeriodicBean();

        int rtn = 0;

        for (WsnQueryMessageBean wqmb : getWsnQueryMessageBeanList()) {
            try {

                executeWsnQueryMessage(wqmb.getQueryId());

                switch (wqmb.getQueryType()) {
                    case DISCONNET: return rtn;
                    case CANCEL:
                        wqmcb =
WsnQueryMessageSqlMap.selectWsnQueryMessageCancel(wqmb.getQueryId());
                        wqmc = new WsnQueryMessageCancel(wqmb.getQueryId());
                        this.ccmb.sendToGateway(wqmc);
                        break ;
                    case ALARM:
                        wqmab = WsnQueryMessageSqlMap.selectWsnQueryMessageAlarm(wqmb.getQueryId());
                        wqma = new WsnQueryMessageAlarm(wqmb.getQueryId(),
                                wqmab.getSensingMode(),
                                wqmab.getSamplingInterval(),
                                wqmab.getZoneId(),
                                wqmab.getSubZoneId(),
                                wqmab.getLocationHeight(),
                                wqmab.getLocationSemantic(),
                                wqmab.getSpatialAggregation(),
                                wqmab.getAlarmCondition(),
                                wqmab.getAlarmThreshold());
                        this.ccmb.sendToGateway(wqma);
                        break ;
                    case PERIODIC:
                        wqmpb =
WsnQueryMessageSqlMap.selectWsnQueryMessagePeriodic(wqmb.getQueryId());
                        wqmp = new WsnQueryMessagePeriodic(wqmb.getQueryId(),
                                wqmpb.getSensingMode(),
                                wqmpb.getSamplingInterval(),
                                wqmpb.getZoneId(),
                                wqmpb.getSubZoneId(),
                                wqmpb.getLocationHeight(),
                                wqmpb.getLocationSemantic(),
                                wqmpb.getSpatialAggregation(),
                                wqmpb.getReportInterval(),
                                wqmpb.getDataProcessing());
                        this.ccmb.sendToGateway(wqmp);
                        break ;
                    case ALL:
                        wqmga = new WsnQueryMessageGetAll(wqmb.getQueryId());
                        this.ccmb.sendToGateway(wqmga);
                        break ;
                }

                rtn++;
```

```
                } catch (java.util.zip.DataFormatException dfe) {
                    System.out.println("Invalid message");
                    rtn = -1;
                }
            }

            return rtn;
        }
}
```

List 7-8. Agent-based Service Integrator Specification

7.4.2.2 Service Resource Description

Agents intend to obtain sensed physical environment information from the sensor field. In order for clients to obtain sensed data from the sensor field, they must first be aware of the available services – thus the need for service discovery. This is a 2-step process.

Step 1

Client queries the WSN for types of sensed data available in a particular location.

Query: QuerySensedDataTypes (location)

Expected response: list of SensedDataType for the specified location where,

location is usually an identifier for the zone that a client is interested in.

SensedDataType are types of sensed information such as temperature, humidity, window contact, or alternative sourced information such as electricity usage.

Step 2

Using the SensedDataType information from Step 1, the client then finds out the capabilities of the WSN in a particular location in terms of a particular SensedDataType. Capabilities are demarcated by the position that they occupy in the location.

The LoginUser.wsdl file is the XML document that describes the LoginUser service interface. It should be placed in the schema/examples/LoginUserService project folder.

The service interface describes how the outside world can interact with this service, specifically the operations that can be performed on it. List 7-9 shows the WSDL code of the LoginUserService.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="LoginService"
targetNamespace= http://examples.strl.org/LoginUser/LoginUserService

xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://examples.strl.org/LoginUser/LoginUserService"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wslw="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.wsdl"
xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd"
xmlns:wsrpw="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreprocessor"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<wsdl:import
      namespace=
      "http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
      location="../../wsrf/properties/WS-ResourceProperties.wsdl" />

<!== Types == >
<types>
<xsd:schema targetNamespace="http://examples.strl.org/usermanagement/LoginService"
xmlns:tns="http://examples.strl.org/ usermanagement/LoginService"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

       <xsd:import
            namespace="http://schemas.xmlsoap.org/ws/2004/03/addressing"
            schemaLocation="../../ws/addressing/WS-Addressing.xsd" />

<!== Requests and Responses == >
       <xsd:element name="add" type="xsd:int"/>
       <xsd:element name="addResponse">
             <xsd:complexType/>
       </xsd:element>

       <xsd:element name="subtract" type="xsd:int"/>
       <xsd:element name="subtractResponse">
             <xsd:complexType/>
       </xsd:element>

       <xsd:element name="getValueRP">
             <xsd:complexType/>
       </xsd:element>
       <xsd:element name="getValueRPResponse" type="xsd:int"/>

<!== Resources Properties == >

       <xsd:element name="LegacyBuildingControlSystem" type="xsd:string"/>
       <xsd:element name="Usermanagement" type="xsd:string"/>
       <xsd:element name="LoginUser" type="xsd:string"/>
       …….
       <xsd:element name="LoginResourceProperties">
             <xsd:complexType>
                   <xsd:sequence>
                         <xsd:element ref="tns:Value" minOccurs="1" maxOccurs="1"/>
                         <xsd:element ref="tns:LastOp" minOccurs="1" maxOccurs="1"/>
                   </xsd:sequence>
```

```
            </xsd:complexType>
        </xsd:element>

</xsd:schema>
</types>

<!== Messages == >
<message name="AddInputMessage">
<part name="parameters" element="tns:add"/>
</message>
<message name="AddOutputMessage">
<part name="parameters" element="tns:addResponse"/>
</message>

<message name="SubtractInputMessage">
<part name="parameters" element="tns:subtract"/>
</message>
<message name="SubtractOutputMessage">
<part name="parameters" element="tns:subtractResponse"/>
</message>

<message name="GetValueRPInputMessage">
<part name="parameters" element="tns:getValueRP"/>
</message>
<message name="GetValueRPOutputMessage">
<part name="parameters" element="tns:getValueRPResponse"/>
</message>

<!== Porttype == >
<portType name="LoginPortType"
wsdlpp:extends="wsrpw:GetResourceProperty"
wsrp:ResourceProperties="tns:CalculatorResourceProperties">

<operation name="add">
<input message="tns:AddInputMessage"/>
<output message="tns:AddOutputMessage"/>
</operation>

<operation name="subtract">
<input message="tns:SubtractInputMessage"/>
<output message="tns:SubtractOutputMessage"/>
</operation>

<operation name="getValueRP">
<input message="tns:GetValueRPInputMessage"/>
<output message="tns:GetValueRPOutputMessage"/>
</operation>

</portType>
</definitions>
```

List 7-9. WSDL Document of LoginUserService

7.4.2.3 Implement Service Using Multi agent

In this case study, the service implementation consists of a single Java class with the code for both the service and the resource. It can also be split into two classes: one for the service and another one for the resource.

The LoginUserService.java file is the service implementation that provides the core functionality for exposing local directory information. It should be placed in the org/strl/examples/services/LoginUser/impl project folder. The source for this Java class is shown as List 7-10.

```
package org.strl.examples.servicesLoginUserimpl;
import java.rmi.RemoteException;
import org.globus.wsrf.ResourceContext;
import org.globus.wsrf.Resource;
import org.globus.wsrf.ResourceProperties;
import org.globus.wsrf.ResourceProperty;
import org.globus.wsrf.ResourcePropertySet;
import org.globus.wsrf.impl.ReflectionResourceProperty;
import org.globus.wsrf.impl.SimpleResourcePropertySet;
import org.strl.examples.LoginUser.LoginUserService.AddResponse;
import org.strl.examples. LoginUser. LoginUser Service.SubtractResponse;
import org.strl.examples. LoginUser. LoginUser Service.GetValueRP;
public class LoginUser Service implements Resource, ResourceProperties
{
        /* Resource Property set */
        private ResourcePropertySet propSet;

        /* Resource properties */
        private int value;
        private String lastOp;

        /* Constructor. Initialises RPs */
        public LoginUser Service() throws RemoteException {
                /* Create RP set */
                this.propSet = new SimpleResourcePropertySet(CalculatorQNames.RESOURCE_PROPERTIES);

                /* Initialise the RPs */
                try {
                        ResourceProperty valueRP = new ReflectionResourceProperty(LoginUser QNames.RP_VALUE,
                "Value", this);
                        this.propSet.add(valueRP);
                        setValue(0);
                        ResourceProperty lastOpRP = new ReflectionResourceProperty(
                        LoginUser QNames.RP_LASTOP, "LastOp", this);
                        this.propSet.add(lastOpRP);
                        setLastOp("NONE");
                } catch (Exception e) {
                                throw new RuntimeException(e.getMessage());
                }
         }

        /* Get Setters for the RPs */
        public int getValue() {
                return value;
        }
```

```
        public void setValue(int value) {
                this.value = value;
        }
        public String getLastOp() {
                return lastOp;
        }
        public void setLastOp(String lastOp) {
                this.lastOp = lastOp;
        }

        /* Remotely-accessible operations */
        public AddResponse add(int a) throws RemoteException {
                value += a;
                lastOp = "ADDITION";
                return new AddResponse();
        }

        public SubtractResponse subtract(int a) throws RemoteException {
                value -= a;
                lastOp = "SUBTRACTION";
                return new SubtractResponse();
        }
        public int getValueRP(GetValueRP params) throws RemoteException {
        return value;
        }

        /* Required by interface ResourceProperties */
        public ResourcePropertySet getResourcePropertySet() {
                return this.propSet;
        }
}
```

List 7-10. Service Implementation Document of LoginUserService

7.5 Experimentation of Software Toolkits

The proposed agent-based service-oriented evolution approach contains many activities, so the support tools are a comprehensive tool suite. In addition, according to the feature of legacy systems, new tools may need to be adopted or developed.

7.5.1 Eclipses

When designing these main tools to support the proposed approach, Eclipse is chosen as a platform to integrate all related tools. Eclipse is an open development platform, as shown in Figure 7-7. The Eclipse platform has a small micro kernel. Except this small micro kernel, everything is written as a plug-in. Each plug-in is loaded by this small micro kernel. Developers can extend Eclipse by writing extensions to the predefined

extension points. Because the openness of Eclipse platform, many tools have been developed as Eclipse plug-ins. Using the Eclipse platform opens the potential to link the tool with a network of other Eclipse plug-in contributions and aims to simplify the number of different, bespoke tools used in software engineering as a whole. For example, a commercial BPEL4WS graphical editor based on Eclipse can be used to replace the basic XML editor I developed.

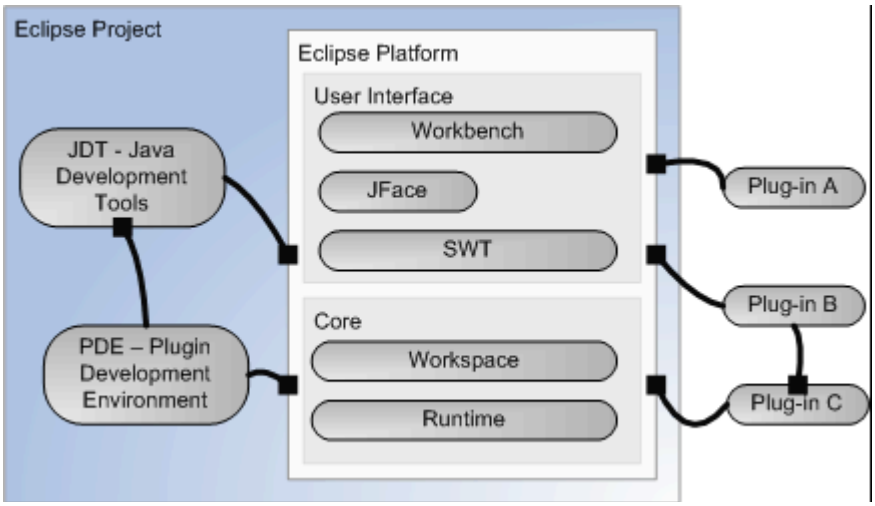


Figure 7-7. Eclipse Plug-in Architecture

Eclipse was originally developed by IBM as the successor of its VisualAge family of tools. Eclipse is now managed by the Eclipse Foundation, an independent not for profit consortium of software industry vendors. It is a free software/open source platform-independent software framework for delivering what the project calls “rich-client applications”, as opposed to “thin client” browser-based applications. So far this framework has typically been used to develop Integrated Development Environments (IDE), such as the Java IDE called Java Development Toolkit (JDT) and compiler that comes as part of Eclipse (and which are also used to develop Eclipse itself).

Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research

institutions and individuals extend, complement and support the Eclipse platform.

The Eclipse tool integration platform is a very popular, very extensible, well-documented IDE that can be configured to host all of the useful development activities from coding to deployment to debugging. The Eclipse IDE can be used to manage all of these artifacts within a single project abstraction and coordinate all of the useful development activities from coding to deployment to debugging. This thesis will use its Java project abstraction to manage the artefacts, such as: source files, WSDLs, client and server stubs, deployment configuration files, etc.

7.5.2 FermaT

Maintainer's Assistant has evolved into an industrial-strength re-engineering tool, FermaT [118, 119, 135], which allows transformations and code simplification to be carried out automatically. The FermaT tool was also designed to use WSL and has applications in the following areas:

- Improving the maintainability of existing mission-critical software.
- Translating programs into modern programming languages. FermaT often translates program written in obsolete assembler language to more modern languages such as C.
- Extracting reusable components from the current system, deriving their specifications, and storing the specifications, implementation, and development strategy.
- Reverse engineering existing systems to high-level specifications, followed by subsequent re-engineering and evolutionary development.

7.5.3 Web Services Wrapper (WSW)

The above method has been tried by the prototype tool Web Services Wrapper (WSW) tool [48], which creates an entry and an interface to the extracted component. WSW is

designed for .Net platform and is a wrapper used to expose interfaces over componentised legacy applications as well as provide meta-data descriptions of legacy data. A wrapped legacy system can function as an autonomous component and may cooperate with other components.

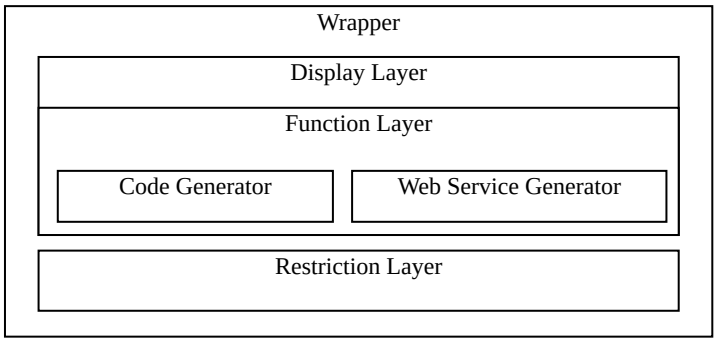


Figure 7-8. Architecture of Web Services Wrapper

Wrapping the legacy application to Web Services need not only to write plenty of wrapping code, but also need to have a strong knowledge of the original system, such as its structure, its behaviour, and its interface. Figure 7-8 shows the architecture of WSW. The wrapper is divided into three layers: Display Layer, Function Layer and Restriction Layer.

The *Display Layer* of the wrapper is mainly used to display the Web Services code and the *Wrapping Report* that is generated by the *Function Layer*. The *Function Layer* of the wrapper implements the wrapping code of Web Services. It contains *Code Generator* and *Web Services Generator*. *Code Generator* generates the Web Services implementation code according to the properties of Web Services and Web Services method, which are set by the developers. *Web Services Generator* calls the compiler of Microsoft .Net to compile the Web Services implementation code and then deploys the Web Services. *Restriction Layer* declares the restrictions of wrapping. Before generating the Web Services, the *Web Services Generator* will validate whether the wrapping methods satisfy these restrictions. Currently, four restrictions are defined:

- **Restriction 1:** *the type of the method must be public.*

Only public methods are useful to be implemented as Web Services methods.

- **Restriction 2:** *abstract methods cannot be wrapped into Web Services method.*

The access of the Web Services method must arouse the execution of the method. The abstract methods, which only define the framework of methods without the real business logic, are not able to be implemented as Web Services methods.

- **Restriction 3:** *overload methods must have different names of Web Services methods.*

Overload methods mean that two or more methods have the same method name, but the parameters are different. If the overload methods should be wrapped into Web Services, they should have different names.

- **Restriction 4:** *if the original method contains transaction and it is not the root object of the transaction, the method cannot be wrapped into the Web Services method.*

The goal of transaction is to maintain the data integrity. All the update operations should be success entirely, or else should be failure entirely. Only the methods which initiate the transaction should be wrapped into Web Services.

Figure 7-9 shows the main form of WSW. On the basis of the analysis of the legacy application, developers decide which classes can be wrapped into Web Services and which methods can be wrapped into Web Services method according to the needs of integration. Before WSW generates the Web Services and related wrapping code, developers should set the properties of Web Services and Web Services method. The properties of Web Services are service name, description and namespace. The properties of Web Services method are MessageName, CacheDuration, EnableSession, TransactionOption, BufferResponse, and Description etc. With these property settings, WSW can further check whether the selected methods satisfy the wrapping restrictions. Validated Web Services and Web Services methods can be wrapped automatically. List 7-10 shows a slice of generated Web Services code.

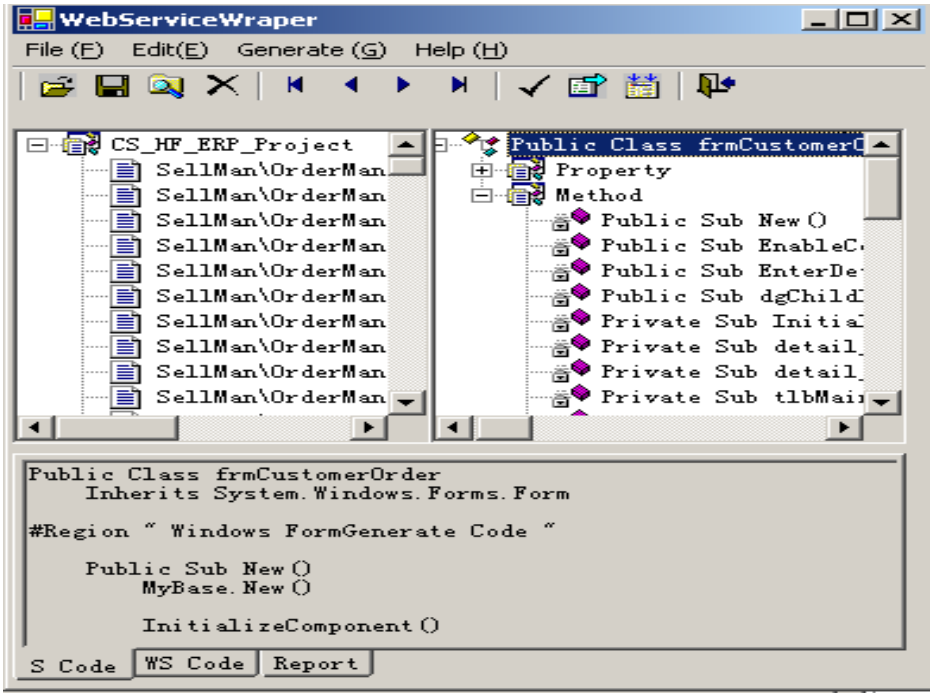


Figure 7-9. Main Form of WSW

```
Imports System.Web.Services
<System.Web.Services.WebService( Description:="SaleOrderService",
  Name:="SaleBill", NameSpace:="URL of the Company")>
  Public Class SaleBill
    InHerits System.Web.Services.WebService
      Public Sub New()
        MyBase.New()
      End Sub
<WebMethod(
  BufferResponse:=True, CacheDuration:="0", Description:="QuerySaleBill",
  MessageName:="QueryBySql",TransactionOption:="Disabled">
  Public Function Query(ByVal sSql As String, ByRef dt As DataSet,
                        Optional ByRef sErrDescr As String="") As Boolean
    ...
End Function
```

List 7-11. Generated Web Services

7.5.4 Tomcat and Sysdeo Tomcat Launcher

Apache Tomcat (formerly under the Apache Jakarta Project) is a Web servlet container developed at the Apache software foundation. Tomcat implements the Java servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a Web server. It adds tools for configuration and management but can also be configured by editing configuration files

that are normally XML-formatted. In this case study, Tomcat is used as the Web service container for the GT4 WSRF Web application.

The Sysdeo Tomcat Launcher is an Eclipse plug-in necessary for managing the Tomcat Web container from the Eclipse IDE. Although there are several Tomcat plug-ins for Eclipse on the market, the (free) Sysdeo Tomcat Launcher is ostensibly the most popular and well known. The Sysdeo Tomcat Launcher adds several menu-bar buttons to Eclipse for the starting and stopping of the embedded Tomcat container, and also registers the Tomcat process with the Eclipse debugger.

7.6 Summary

The selected case study was used to demonstrate that the agent-based service-oriented approach can help to evolve legacy systems into pervasive computing environment systematically and cost-effectively. The case study is designed to investigate specific research questions, focusing on different claims.

- Legacy system analysis with program transformation, program slicing and software clustering.
- Pervasive services integration with multi-agent architecture includes services stateful resources description, services implementation, deployment parameters definition and services deployment.

Chapter 8
Conclusions

In this chapter, the strengths and significance of major results are summarised, an evaluation is provided by revisiting the questions, hypothesis, contributions and success criteria put forward in Chapter 1, the weaknesses and limitations of the work are discussed, and finally, areas for future work are identified.

8.1 Summary

In the context of software evolution, legacy software systems are continuously evolved in order to correct errors, provide new functionality, or port into modern platforms. Pervasive computing is an increasingly popular topic, which emerged within a project designed to link wide-area supercomputing resources to computational and collaborative science. With the adoption of the service oriented architecture, Pervasive computing services are emerged by integrating Pervasive computing technologies and Web services to perform a seamless information processing system across distributed, heterogeneous, dynamic virtual organisations that the user can access from any location. Could legacy software systems be evolved into Pervasive computing environment? The answer is "Yes". The proposed approach in this thesis integrates traditional methods with those emerging technologies:

- Component based development. Component based software engineering is a process that aims to design and construct software systems using reusable software components. It emphasis on decomposition of the engineered systems into functional or logical components with well defined interfaces used for communication across the components.

- Pervasive computing services. Pervasive service have emerged by combining Web services and Pervasive computing to perform a seamless information processing system across distributed, heterogeneous, dynamic virtual organisations that the user can access from any location.

- Software reverse engineering techniques. In this thesis, program transformation and program slicing technique is used to decompose system, understand program, eliminate dead code and make selected code segments function independently by component interface parameters determination and deep source code comprehension. The clustering analysis technique is used to group large mounts of entities in a dataset and capture reusable legacy code segments into clusters according to their relationship and similarity from legacy systems, and create a hierarchical structure of these reusable legacy code segments.

As Pervasive computing is an emerging technique, there is not much work for integrating Pervasive computing technique with software evolution. From the legacy systems evolution perspective, few of them bridge software evolution and Pervasive computing development together. Most of these researches focus on the reusable resources identification from legacy systems, but they can not be integrated in the Pervasive computing system directly. Most current research works of service oriented software evolution are still stay on Web services.

Even though some researches focus on the Pervasive computing oriented evolution areas, they only concentrate on parts of reengineering process and do not provide a general framework of the legacy systems evolution towards Pervasive computing environment. Moreover, some of them are even based on the unfashionable standards such as Open Pervasive computing Services Architecture and Open Pervasive computing Services Infrastructure. Generally speaking, current approaches about reengineering legacy systems with Pervasive computing technology are not mature and accomplished, novel approaches are required in this research area.

In this thesis, an agent-based service-oriented approach is proposed to evolving legacy software evolution into Pervasive computing environment. The proposed

approach employs reverse engineering techniques to recover service from legacy systems and deploys these services in agent-based service-oriented architectures.

This agent-based service-oriented reengineering methodology is novel and feasible. Because the complete architecture recovery of legacy systems is difficult to archive and the cost of reengineering the whole legacy system is expensive, reengineering legacy systems by reusing recovered legacy components is economic and efficient. The combination of program transformation and program slicing techniques successfully analyses the legacy code at different level and ascertains the efficiency of extracted reusable legacy code.

In a word, complex legacy systems can be decomposed into functions and components. This functional code can be isolated and fashioned into software services, which can be integrated in agent-based service-oriented architectures. This approach is modestly invasive but offers a high return on investment for legacy assets and many benefits for software service development. This agent-based service-oriented software reengineering approach offers more flexibility, better system understanding, easier maintenance and reduced costs. It is in many ways a significant approach to migrate from legacy systems to service-oriented architectures and benefit the construction and orchestration of Web services.

Affected by the Pervasive Computing trend, many existing software systems will turn into legacy systems. These legacy systems require to be evolved into Pervasive Computing environment, which can facilitate legacy system evolution in Pervasive computing service orientated architecture. Comparing design a new system, evolving legacy system into Pervasive computing environment could massively reduce time and cost for the enterprise. And it will bring more flexibility, expansibility and reusability.

Pervasive computing oriented legacy software system evolution is a challenge in the new era of software engineering. This thesis develops an effective approach to evolve legacy software systems into Pervasive computing environment. In particular, first, reverse engineering techniques are used for program comprehension and design recovery. Then the legacy software systems are decomposed into a hierarchy of

subsystems by defining relationships between the entities of the underlying paradigm of the legacy system. The decomposition is driven by program slicing and clustering techniques. Next, Pervasive computing components are created by wrapping objects and defining the interface. Finally, Pervasive computing components are allocated to Pervasive computing services environment by specifying the requirements of the system and characteristics of the network as an integer programming model. The aim of this approach is to use legacy systems into Pervasive computing environment which enables the integration of legacy resources with Pervasive computing across distributed, heterogeneous, dynamic environment and communities. Around this theme, several significant technical issues were addressed:

Legacy system decomposition and service identification. Because the complete reverse engineering of legacy systems is difficult to archive and the cost of reengineering the whole legacy system is expensive, to reuse recovered legacy components is an economic and efficient way to reengineer legacy systems. This component based Pervasive computing oriented reengineering methodology is feasible, and it contributes on reengineering legacy software systems into Pervasive computing services environment. This approach is modestly invasive, but it offers a high return on investment for legacy assets and many benefits for the Pervasive computing service development.

XML representation and Pervasive computing service composition. Once a software component has been extracted from a legacy system, or has been built as a new component, its interface can be extracted and represented in XML. The XML representation is not only finished by the component wrap, but also finished with the sources code analysis included such as the using of AST, DTD and XSLT. The term “Pervasive service” refers to a collection of legacy reusable resources. It describes the relation of legacy resources and the semantic rules governing the resources. A Pervasive service is theoretically independent of its representation schema. The resulting service with core legacy code function in Pervasive service framework for the intent of sharing distributed resources and coordinated problem solving.

Pervasive computing service integration. To achieve the Pervasive computing

service integration, different from the related studies, the proposed approach describes a method for defining the legacy resources as stateful resources, and builds the Pervasive computing services based on these reusable resources. Four steps are summarised to migrate Pervasive computing components in the new Pervasive computing services environment. In the first step, the service's properties and its interface are defined by WSDL. Then the implementation of the service is carried out by Java. In the third step, the WSDD and JNDI file define the deployment parameters include service registration and resources localisation. The last step is to deploy the Pervasive computing service. This Pervasive computing service oriented reengineering methodology brings more flexibility, expansibility and reusability, as well as more reliability. All in all, legacy systems can be extracted and reengineered into stateful resources by Pervasive computing oriented evolution approach. These stateful resources can be isolated and integrated into Pervasive computing service architectures. The proposed approach contributed on evolving a legacy software system into Pervasive computing services by an improved reverse engineering method.

8.2 Contributions and Evaluation

In this thesis, an Agent-based Service-Oriented Approach to Evolving Legacy System into a Pervasive Computing Environment is proposed, which integrates all technical supports into a systematic method for software evolution in Pervasive Computing environment. As observed in Chapter 1, Specifically in Chapter 4, the original contributions of this thesis are as follows:

C1: In Chapter 4, a unified approach integrates the software evolution approach with the Pervasive Computing technique. It utilises reusable legacy resources into Pervasive Computing environment to build pervasive applications across distributed, dynamic environment and service oriented architecture communities. This research develops an effective way to reuse legacy assets with Pervasive Computing technology.

C2: In Chapter 5, identifying service from legacy software systems for the use in

Pervasive Computing environment with reverse engineering techniques such as program slicing, software clustering and programme transformation technologies.

C3: In Chapter 5, using the service-oriented design approach into the Pervasive Computing environment.

C4: In Chapter 6, an evolvable agent-based service-oriented architecture can integrate the service which identified from legacy system into the Pervasive Computing environment.

C5: In Chapter 6, a Multi-agent architecture and transformation method that transforms between legacy files and XML documents. The representation is finished not only by the component wrap, but also with the sources code analysis included such as the using of AST, DTD and XSLT.

8.3 Conclusions

Pervasive computing is a new technology, so there is currently few active research related to evolve legacy systems within Pervasive computing environment. Most researches are necessary and significant in this area to leverage and extend legacy resources in Pervasive computing environment. From economic aspects, a business is constantly re-organising, changing its boundaries and reconfiguring its activities. Pervasive computing oriented evolution enables legacy systems to adapt continuous changing in business logic and market requirements. From technical aspects, legacy applications integration towards Pervasive computing and service will become common in Pervasive computing oriented environment. As a result, there are increasing interests in migrating and reengineering legacy software systems with these new Pervasive computing technologies and software development paradigms.

This research proposes a solution for evolving legacy software systems using a Pervasive computing user interface to enable the dynamic activation and Pervasive computing resources discovery. This ability of the legacy system to use Pervasive computing will allow the enterprise to take advantage of the broad commercial support

provided by Pervasive Computing technology.

Concerning on the reuse of legacy software system, because the complete recovery of legacy systems is difficult to archive and the cost of reengineering the whole legacy system is expensive, reusing is an economic and efficient way to reengineer legacy systems.

Through this research experience, it is argued that the detailed component mining approach needs to be tailored according to the features of a particular legacy system. The adopted software clustering techniques and program slicing techniques are not new, but the comprehensive analysis ascertains the reusable legacy code efficiently.

Through the discussion in this thesis, it is concluded that the legacy system evolution can assist Pervasive computing application development. The proposed legacy system evolution framework is powerful for utilising reusable legacy resources into Pervasive computing environment to build Pervasive computing applications across distributed, dynamic environment and service oriented architecture communities.

Based on the process of system decomposition, resources representation and Pervasive computing environment integration presented in this thesis, the legacy system assets can be successfully evolved into the Pervasive computing environment. Comparing to design a brand new system, based reusing legacy systems, this agent-based service-oriented evolution approach is feasible and it massively reduce the developing time and cost for the enterprise.

8.4 Limitation

The difficulties in the evolution of legacy systems with Web services and SOAs should not be underestimated. Every legacy system offers unique technical challenges and every organisation has unique needs, so the best legacy extension methods vary case by case. In fact, there is considerable resistance to modernising legacy systems, due to the immaturity of the technology used and the uncertain business value of such a costly migration effort.

Furthermore, it is important to note that legacy systems will never have native Web services interfaces. The existing legacy systems must be bridged into Web services. The latency involved in each call would likely be a barrier to successful integration. Software reengineering is not a panacea, and therefore, it will not solve all the problems of the software maintenance crisis. It is a promising starting point towards establishing good standards for producing maintainable software.

Finally, the performance issue with Web services needs to be considered. There is a significant time penalty to be paid for interpreting the business process language and in sending messages to and from the application server. Parsing the WSDL requests and responses is only one of the many time consuming bottlenecks. The conversion of the WSDL interface to the local language interface and back is minor problem compared to the total performance issue. Measurements have shown that it will not consume more than 15% of the total time in processing transactions. Relative to interpreting the BPEL, organising and dispatching the messages and parsing the WSDL interfaces it is insignificant. It must be seen that the dynamic binding of business processes to distributed Web services is in itself a resource consuming process with or without wrapping. Thus, it can be concluded that using wrapped Web services is hardly less efficient than using non-wrapped Web services [109].

8.5 Future Work

The future work of this research may include the following issues:

- Infrastructure and Framework. For the research, the implementation of the proposed framework is enough to demonstrate the Pervasive computing oriented evolution concept. However, the framework need to be developed for meet more type of Pervasive computing applications in the future.

- Legacy system analysis and resource identification. Although a legacy system decomposition and component mining approach are proposed to identification components for using in Pervasive computing environment, there is not enough experiment done due to limited time of this research and it may not suitable for all

kinds of legacy systems. For evolving full-blown legacy software systems, they need to be further developed.

- The autonomic computing can be introduced to establish a framework for triggering Website reengineering process by an automatic perception of changes in the business requirements and environments.

- The section of semantic Pervasive computing oriented evolution focuses on how to retrieve the useful resources from legacy systems and represent the resources based on the semantic Pervasive computing standards. In the future, more attention will be paid on how to run the reusable resources in the semantic Pervasive computing environment, which may include the following aspects: Expose the meaning of Pervasive computing services, resources and entities by assertions in a common RDF data model. Publish and share consensually agreed ontology in Web ontology language OWL and query, filter, integrate and aggregate the metadata in RDF Data Query Language (RDQL).

References

[1] "The Endeavour Expedition," http://endeavour.cs.berkeley.edu/, University of California, Berkeley.

[2] "Jini™ Architectural Overview," Technical White Paper, Sun Microsystems. Inc.

[3] "Oxygen at the Massachusetts Institute of Technology," Laboratory for Computer Science (LCS), http://www.lcs.mit.edu/news/scifi.html.

[4] "Planet Blue at IBM Research," http://www.research.ibm.com/compsci/planetblue.html.

[5] "Portolano at the University of Washington," http://portolano.cs.washington.edu/.

[6] "Project Aura," Carnegie Mellon University, http://www.cs.cmu.edu/~aura/.

[7] *IEEE Std. 1219: IEEE Internet Computing*. Los Alamitos CA., USA.: IEEE Computer Society Press, 1993.

[8] "Information Technology Software Lifecycle Processes," ISO12207, International Standards Organisation, Geneva, Switzerland 1995.

[9] "Web Services Architecture," http://www.w3.org/TR/ws-arch/. W3C Working Group Website, 2004.

[10] A. Brown, S. Johnston and K. Kelly, "Using Service-oriented Architecture and Component-based Development to Build Web Service Applications," Rational Software Corporation, Tech. Rep 2002.

[11] A. Yeh, D. Harris and H. Reubenstein, "Recovering Abstract Data Type and Object Instances from a Conventional Procedural Language," Working Conference on Reverse Engineering (WCRE), 1995.

[12] J. Hwang and P. Aravamudham, "Middleware Services for P2P Computing in Wireless Grid Networks," *IEEE Internet Computing*, vol. 18, pp. 40-46, 2004.

[13] N. Shankar and W. Arbaugh, "On Trust for Ubiquitous Computing," in *Ubicomp*,

2002.

[14] B. Korel and J. Rilling, "Dynamic Program Slicing Methods," *Information and Software Technology*, vol. 40, pp. 647-659, 1998.

[15] B. Qiao, H. Yang, W. Chu and B. Xu, "Bridging Legacy Systems to Model Driven Architecture," 27th Annual International Computer Software and Application Conference (COMPSAC), 2003.

[16] L. Barkhuus, "Ubiquitous Computing: Transparency in Context-aware Mobile Computing," in *UbiComp* Doctoral Consortium, Gothenborg, 2004.

[17] P. Baumann, J. Fassler, M. Kiser, Z. Ozturk, and L. Richter, "Semantics-based Reverse Engineering," Technical Report 94.08, Department of Computer Science, University of Zurich, Switzerland 1994.

[18] H. Bergsten, *Java Server Page*: O'Reilly Publishing, 2003.

[19] G. Borriello, "Key Challenges in Communication for Ubiquitous Computing," *IEEE Communication, 50th Anniversary Issue*, 2002.

[20] F. P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, vol. 20(4), pp. 10-19, Apr. 1987.

[21] T. Bull, "An Introduction to the WSL Program Transformer," Technical Report, University of Durham, England 1995.

[22] M. Burger, "The Path Towards Pervasive Computing - A Network Approach," in *O'REILLY Emerging Technology Conference*, 2002.

[23] C. Nester, M. Philippsenand and B. Haumacher," A More Efficient RMI for Java," in *ACM conference on Java Grande*, 1999.

[24] C. Chiang, "The Use of Adapters to Support Interoperability of Components for Reusability," *Information and Software Technology*, vol. 45, pp. 149-156, 2003.

[25] E. M. Clarke and J. M. Wing, "Formal Methods: State of the Art and Future Directions," *ACM Computing Surveys*, vol. 28(4), pp. 626-643, Sep. 1996.

[26] G. Heineman and W. Councill, "Component Based Software Engineering: Putting the Pieces Together," *ACM Press*, 2001.

[27] E. Chikofsky and J. Cross, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, 1990.

[28] E. Chikofsky and J. Cross, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, pp. 13-17, 1990.

[29] D. Alur, J. Crupi and D. Malks, *Core J2EE Patterns*. NJ, USA: Prentice Hall PTR, Upper Saddle River, 2001.

[30] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, "Web Services Architecture," W3C Working Draft, Available at: http://www.w3.org/TR/2003/ WD-ws-arch-20030514/#id2608426., 2003.

[31] D. Gorissen, P. Wendykier, D. Kurzyniec and V. Sunderam, "Integrating Grid Information Services with JNDI," in *15th International Heterogeneous Computing Workshop (HCW)*. Greece, 2006.

[32] D. Kuck, R. Kuhn, D. Padua, B. Leasure and M. Wolfe, "Dependence Graphs and Compiler Optimisations," 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Virginia, USA, 1981.

[33] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu, "Peer-to-Peer Computing," Technical Report HPL-2002-57, HP Laboratories 2002.

[34] D. Quercia, S. Hailes and L. Capra, "MobiRate: Making Mobile Raters Stick to their Word," in *ACM Ubicomp*, 2008.

[35] N. Davies and H.-W. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems," *Pervasive Computing, Mobile and Ubiquitous Systems*, vol. 1(1), pp. 26-35, 2002.

[36] C. Dhawan, *Mobile Computing, A System Integrator's Handbook*: McGraw-Hill, 1997.

[37] T. Downing, *Java RMI: Remote Method Invocation*. Foster City, CA, USA: IDG Books Worldwide, 1998.

[38] R. Koschke and T. Eisenbarth, "A Framework for Experimental Evaluation of Clustering Techniques," 8th International Workshop on Program Comprehension

(IWPC), Limerick, Ireland, 2000.

[39] G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web Services Concepts, Architectures and Applications*: Springer Verlag, 2004.

[40] G. Canfora, A. Cimitile, A. De Lucia and G. Di Lucca, "Decomposing Legacy Programs: A First Step Towards Migrating to Client-Server Platforms," *Journal of Systems and Software*, vol. 54, pp. 99-110, 2000.

[41] G. Di Lucca, A. Fasolino, F. Pace, P. Tramontana and U. DeCarlini, "Comprehending Web Applications by a Clustering Based Approach," 10th International Workshop on Program Comprehension (IWPC), 2002.

[42] D. Binkley and K. Gallagher, "Program Slicing," *Advances in Computers*, vol. 43, pp. 1-50, 1996.

[43] R. Grimes and R. Grimes, *Professional DCOM Programming*. Birmingham, UK: Wrox Press Ltd., 1997.

[44] R. Grimm, T. Anderson, B. Bershad and D. Wetherall, "A System Architecture for Pervasive Computing," 9th ACM SIGOPS European Workshop, Denmark, 2000.

[45] R. Grimm, T. Anderson, B. Bershad and D. Wetherall, "Programming for Pervasive Computing Environments," Department of Computer Science and Engineering, University of Washington 2001.

[46] D. Atkinson and W. Griswold, "The Design of Whole Program Analysis Tools," in *International Conference on Software Engineering (ICSE)*, 1996.

[47] D. Atkinson and W. Griswold, "Implementation Techniques for Efficient Data-Flow Analysis of Large Programs," International Conference on Software Maintenance (ICSM), 2001.

[48] H. Guo, C. Guo, F. Chen and H. Yang, "Wrapping Client-Server Application to Web Services for Internet Computing," 6th IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05), Dalian, China, Dec. 2005.

[49] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish and H. Duman,

"Creating an Ambient-Intelligence Environment Using Embedded Agents," *IEEE Intelligent Systems*, vol. 19, pp. 12-20, 2004.

[50] H. Ryu, G. Hong and H. James, "A Research on Quality Assessment Technique for Ubiquitous Software or Middleware," Massey university, New Zealand 2004.

[51] X. Liu, H. Zedan and H. Yang, "Abstraction: A Key Notion for Reverse Engineering in A System Re-engineering Approach," *Software Maintenance and Evolution: Research and Practice*, vol. 12, pp. 197-228, 2000.

[52] P. Thiran and J. Hainaut, "Wrapper Development for Legacy Data Reuse," 8th IEEE Working Conference on Reverse Engineering, Germany, 2001.

[53] K. Henricksen, J. Indulska and A. RAkotonirainy, "Infrastructure for Pervasive Computing: Challenges," in *Workshop on Pervasive Computing and Information Logistics at Informatik* Vienna, 2001.

[54] V. Tzerpos and R. Holt, "MoJo: A Distance Metric for Software Clustering," 6th Working Conference on Reverse Engineering (WCRE), Atlanta, 1999.

[55] N. Howarth, "Abstract Syntax Tree Design," APM.1551, APM Ltd., Cambridge, U.K 1995.

[56] N. Howarth, "Rules for Type Inferencing," APM.1552, APM Ltd., Cambridge, U.K 1995.

[57] M. P. Singh and M. N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*: John Wiley and Sons, 2005.

[58] J. Hunter, *Java Servlet Programming*: O'Reilly Publishing, 2001.

[59] I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," *Lecture Notes in Computer Science*, vol. 2735, pp. 118-128, 2003.

[60] IEEE, "IEEE Standard Collection: Software Engineering," IEEE Inc., New York, 1997.

[61] IO, "Application Modernisation and Legacy-to-SOA," Interactive Objects, http://www.interactive-objects.com/solutions/application-modernisation/.

[62] Y. Irene, L. Chen, J. Stephen, H. Yang and J. Zhang, "Ubiquitous Provision of Context Aware Web Services," IEEE International Conference on Services Computing (SCC'06), 2006.

[63] J. Chung, K. J. Lin and R. G. Mathieu, "Web Services Computing: Advancing Software Interoperability," *IEEE Computer*, vol. 36, pp. 35-37, 2003.

[64] J. Ferrante, K. Ottenstein and J. Warren, "The Program Dependence Graph and Its Use in Optimisation," *ACM Transactions on Programming Languages and Systems*, vol. 19, 1987.

[65] J. Siegel, S. Baker, M. Balick, D. Frantz and H. Mirsky, *COBRA Fundamentals and Programming*. NY, USA: John Wiley & Sons, 1996.

[66] B. Jacob, "Grid Computing: What Are the Key Components," IBM DeveloperWorks 2003.

[67] C. Johnson, "Basic Research Skills in Computing Science," Department of Computer Science, Glasgow University, UK.

[68] K. Arnold, R. Scheifler, J. Waldo, A. Wollrath and B. O'Sullivan, *A Jini Specification*. Boston, MA, USA: Addison Wesley Longman Publishing Company, 1999.

[69] K. Gallagher and J. Lyle, "Using Program Slicing in Software Maintenance," *IEEE Transactions on Software Engineering*, vol. 17, pp. 751-761, 1991.

[70] A. Deursen and T. Kuipers, "Identifying Objects Using Cluster and Concept Analysis," 21th International Conference on Software Engineering (ICSE), Los Angeles, USA, 1999.

[71] L. Bent, D. Atkinson and W. Griswold, "A Comparative Study of Two Whole-Program Slicers for C," Technical Report CS2000-0643, University of California at San Diego, USA 2001.

[72] M. Ladkau, F. Chen, S. Natelberg and S. Li, "FermaT Transformation Engine Tutorial," Technical Report, STRL, Department of Computer Science, De Montfort University, UK, Nov. 2006.

[73] J. Li, "A Novel Approach to Evolving Legacy Software Systems into a Grid

Computing Environment," PhD Thesis, De Montfort University, England, 2007.

[74] X. Liu, "Abstraction: A Notation for Reverse Engineering," PhD Thesis, De Montfort University, England, 1999.

[75] M. Krajecki, O. Flauzac and P. Merel, "Focus on the Communication Scheme in the Middleware CONFIIT using XML-RPC," 18th International Parallel and Distributed Processing Symposium (IPDPS), 2004.

[76] M. Mock, D. Atkinson and S. Eggers, "Program Slicing with Dynamic Points-to Sets," *IEEE Transactions on Software Engineering*, vol. 31, pp. 657-678, 2005.

[77] M. Serrano, D. Carver and C. Montes, "Reengineering of Legacy Systems to Distributed Object Environments," *Systems and Software*, vol. 64, pp. 37-55, 2002.

[78] M. Turner, D. Budgen and P. Brereton, "Turning Software into a Service," *Computer*, vol. 36, pp. 38-44, 2003.

[79] R. C. Millham, "Evolution of Batch-Oriented COBOL Systems into Object-Oriented Systems through Unified Modelling Language," PhD Thesis, De Montfort University, England, 2006.

[80] R. C. Millham, J. Pu, and H. Yang, "TAGDUR: A Tool for Producing UML Sequence, Deployment, and Component Diagrams Through Reengineering of Legacy Systems," IASTED Conference on Software Engineering, Innsbruck, Austria 2004.

[81] R. C. Millham and H. Yang, "TAGDUR: A Tool for Producing UML Diagrams Through Reengineering of Legacy Systems," IASTED Conference on Software Engineering, Marina del Ray, USA, 2003.

[82] D. Millicev, "Domain Mapping Using Extended UML Object Diagrams," *IEEE Software*, pp. 90-97.

[83] R. Monson-Haefel, *Enterprise JavaBeans*: O'Reilly Publishing, 2001.

[84] G. D. Abowd and E. D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing," *ACM Trans. Computer-Human Interaction*, vol. 7(1), pp. 29-58, 2000.

[85] A. Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Sebastopol, CA: O'Reilly & Associates, 2001.

[86] P. Huy, K. Takahiro and H. Tetsuo, "Web Service Gateway - A Step Forward to E-Business," IEEE International Conference on Web Services (ICWS), 2004.

[87] D. L. Parnas, "Designing Software for Ease of Extension and Contraction," *IEEE Transaction on Software Engineering*, vol. 5(2), pp. 128-138, March 1979.

[88] P. Gupta and D. Moitra, "Evolving a Pervasive IT Infrastructure: a Technology Integration Approach," *Personal and Ubiquitous Computing, ACM*, vol. 8(1), 2004.

[89] R. C. Seacord, D. Plakosh and G. A. Lewis, *Modernising Legacy Systems*: Addison-Wesley Press, 2003.

[90] R. Engelen, G. Gupta and S. Pant, "Developing Web Services for C and C++," *IEEE Internet Computing*, vol. 7, pp. 53-61, 2003.

[91] R. Orfali and D. Harkey, *Client/server Programming with Java and CORBA*. N Y, USA: John Wiley & Sons, 1998.

[92] R. S. Arnold, *Software Reengineering*: IEEE Computer Society Press, 1993.

[93] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell and K. Nahrstedt, "A Middleware Infrastructure for Active Space," *IEEE Pervasive Computing*, vol. 1, 2002.

[94] H. Romesburg, *Cluster Analysis for Researchers*: Lulu Press, 2004.

[95] S. Rugaber, "A Tool Suite for Evolving Legacy Software," IEEE International Conference on Software Maintenance (ICSM), 1999.

[96] S. Comella-Dorda, K. Wallnau, R. Seacord and J. Robert, "A survey of Black-Box Modernisation Approaches for Information Systems," International Conference on Software Maintenance (ICSM), 2000.

[97] S. Horwitz, T. Reps and D. Binkley, "Interprocedural Slicing Using Dependence Graphs," *ACM Transactions on Programming Languages and Systems*, vol. 12, pp. 26-60, 1990.

[98] S. Jean, V. Chua, P Echevarria and J. M. Mendoza, "Bayanihan Computing .NET: Grid Computing with XML Web Services," in *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, Germany, 2002.

[99] S. Tilley, J. Gerdes, T. Hamilton, S. Huang, H. Muller, D. Smith and K.Wong, "On the Business Value and Technical Challenges of Adopting Web Services," *Software Maintenance and Evolution Research and Practice*, vol. 16, pp. 31-50, 2004.

[100] S. Zweben, S. Edwards, B. Weide and J. Hollingsworth, "The Effects of Layering and Encapsulation on Software Development Cost and Quality," *IEEE Transactions on Software Engineering*, vol. 21, pp. 200-208, 1995.

[101] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, 2001.

[102] R. Schwanke, "An Intelligent Tool for Re-engineering Software Modularity," 13th International Conference on Software Engineering (ICSE), USA, 1991.

[103] R. C. Seacord, D. Plakosh and G. A. Lewis, *Modernising Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*: Addison Wesley, 2003.

[104] R. Sessions, *COM and DCOM: Microsoft's Vision for Distributed Objects,* NY, USA: John Wiley & Sons, 1997.

[105] H. Sneed, "Object Oriented COBOL Recycling," 3rd IEEE Working Conference of Reverse Engineering (WCRE), USA, 1996.

[106] I. Sommerville, *Software Engineering, 5th Edition*: Addison Wesley, 1995.

[107] E. Stroulia, M. El-Ramly and P. Sorenson, "From Legacy to Web through Interaction Modeling," International Conference on Software Maintenance (ICSM'02), Montrzal, Canada, Oct. 2002.

[108] C. Szyperski, *Component Software beyond Object-Oriented Programming*: Addison-Wesley Press, 2002.

[109] M. Thomas, R. Redmond, V. Yoon and R. Singh, "A Semantic Approach to Monitor Business Process Performance," *Communication of ACM*, vol. 48, pp. 55,

2005.

[110] F. Tip, "A Survey of Program Slicing Techniques," *Journal of Programming Languages*, vol. 3, pp. 121-189, 1995.

[111] F. Ricca and P. Tonella, "Using Clustering to Support the Migration from Static to Dynamic Web Pages," 11th IEEE International Workshop on Program Comprehension (IWPC), 2003.

[112] U. Rutishauser, J. Joller and R. Douglas, "Control and Learning of Ambience by an Intelligent Building," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 35, pp. 121-132, 2005.

[113] V. Callaghan, G. Clarke, M. Colley and H. Hagras, "A Soft-Computing DAI Architecture for Intelligent Buildings," *Soft Computing Agents: New Trends for Designing Autonomous Systems*, vol. 75, 2001.

[114] S. Vinoski, "Integration with Web Services," *IEEE Internet Computing*, vol. 7, pp. 75-77, 2003.

[115]W. Vogels, "Web Services are not Distributed Objects," *IEEE Internet Computing*, vol. 7, pp. 59-66, 2003.

[116] J. Ward, "Hierarchical Grouping to Optimise an Objective Function," *Journal of the American Statistical Association*, vol. 58, pp. 236-244, 1963.

[117] M. Ward, "Proving Program Refinements and Transformations," PhD Thesis, Oxford University, England, 1989.

[118] M. Ward, "Program Analysis by Formal Transformation," *Computer Journal*, vol. 39(7), pp. 598–618, 1996.

[119] M. Ward, "The FermaT Assembler Re-engineering Workbench," IEEE International Conference on Software Maintenance (ICSM'01), 2001.

[120] M. Ward, "Program Slicing via FermaT Transformations," 26th IEEE Annual International Computer Software and Applications Conference (COMPSAC'02), Oxford, England, Aug. 2002.

[121] M. Ward and H. Zedan, "MetaWSL and Meta-Transformations in the FermaT Transformation System," 29th IEEE Annual International Computer Software and

Applications Conference (COMPSAC'05), 2005.

[122] M. Ward and H. Zedan, "Slicing as a Program Transformation," *ACM Transactions on Programming Languages and Systems*, vol. 29(2), Mar. 2007.

[123] I. Warren, *The Renaissance of Legacy Systems: Method Support for Software-System Evolution*: Springer-Verlag, 1999.

[124] M. Weiser, "Program Slicing," *IEEE Transactions on Software Engineering*, vol. 10, 1984.

[125]M. Weiser, "The Computer for the Twenty-First Century," *Scientific American*, pp. 94-104, 1991.

[126] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, 1993.

[127] M. Weiser, "Ubiquitous Computing," *IEEE Computer*, 1993.

[128] T. Wiggerts, "Using Clustering Algorithms in Legacy Systems Remodularisation," in *4th Working Conference on Reverse Engineering (WCRE)*. Netherlands, 1997.

[129] X. Liu, H. Yang and H. Zedan, "On the Assessment of Re-engineering Tools," in *International Workshop on Empirical Studies in Software Maintenance (WESS)*. Oxford, UK, 1999.

[130] K. Bennett and J. Xu, "Software services and software maintenance," 7th European Conference on Software Maintenance and Reengineering (CSMR ‘03), 2003.

[131] Y. Huang, I. Taylor, D. Walker and R. Davies, "Wrapping Legacy Codes for Grid-Based Applications," 17th International Parallel and Distributed Processing Symposium (IPDPS), 2003.

[132] Y. Jiang and E. Stroulia, "Towards reengineering Web sites to Web-services providers," 8th European Conference on Software Maintenance and Reengineering (CSMR), 2004.

[133] Y. Maarek, R. Fagin, I. Shaul and D. Pelleg, "Ephemeral Document Clustering for Web Applications," Technical Report RJ 10186, IBM Research 2000.

[134] H. Yang, "The Supporting Environment for A Reverse Engineering System - The Maintainers Assistant," IEEE Conference on Software Maintenance 1991, Sorrento, Italy, Oct. 1991.

[135] H. Yang and M. Ward, *Successful Evolution of Software Systems*: Artech House, Inc., 2003.

[136] Z. Zhang and H. Yang, "Incubating Services in Legacy Systems for Architectural Migration," 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004.

[137] T. Mowbray and R. Zahavi, *The Essential CORBA: Systems Integration Using Distributed Objects*: John Wiley and Sons, 1995.

Appendix A: Source Code of a Building Control System

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*     T F M 1 3   P R O G R A M   -   C O N T R O L   M O D U L E                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

;*******************************************************************
 RAMAD      EQU 9FFFH          ;RAM address 0___9FFFH
    WDCON          EQU 0A5H
    T3         EQU 0FFH
     SPSP       EQU 5AH        ;stack 5BH___7FH
    TMH0   EQU   0CEH                 ;T0 20.833MS
    TML0   EQU   00H
     TMH1       EQU   0FEH            ;T1 RS_485    9600
    TTYL   EQU   58H       ;TYPE CHECK 10 MINUTE 0258H
    TTYH   EQU   02H
    PSTUS  EQU   40H
    BUFFER10 EQU 4900H                ; ms+1;10s+1;s+1(2 byte);s_1;
    BUFFER11 EQU 4910H
    BUFFER12 EQU 4920H
    LSBD      EQU 4909H            ;SCAN BEGINING DELAY TIMER
    DSPT      EQU 4908H    ;DISP BOARD 5 SECOND
;_______________________________BIT FLAG BLOCK
    BIT00   BIT    00H        ;type check
    BIT01   BIT    01H        ;loop card check
     BIT02      BIT    02H          ;stop walk_test
     BIT03      BIT    03H          ;turn on auto_test
     BIT04      BIT    04H          ;continue to auto_test
    BIT05   BIT    05H        ;LED pulse on/off
    E2EOR      BIT    06H          ;E2 WRITE ERROR
    BIT09      BIT    09H     ;LOOP SCAN BEGINING DELAY
    BIT10      BIT 10H
    BIT11   BIT 11H
    BIT12   BIT 12H
    BIT13   BIT 13H
    BIT14   BIT 14H
    BIT15   BIT 15H
    END_BF   BIT 16H                 ;BIT16 END_B TEST TIME_FLAG
    BIT17   BIT 17H
    BIT18   BIT 18H
     C232E      BIT 19H            ;RS_232 ERROR FLAG
    BIT1A   BIT 1AH                  ;0:0:0 FLAG
    BIT1B   BIT 1BH                  ;WALK TEST BIT02
    BIT1C   BIT 1CH
    POF1    BIT 1DH                  ;power error flag
    PTF1    BIT 1EH                  ; REAL TIME PRINT FLAG
     PTF2       BIT 1FH            ; PRINT NO_CONNECTION FLAG
     PTF3       BIT 20H            ;SCANNING INIATATION DELAY
FALG
     PTF4       BIT 21H            ;LOOP CARD 1 INSTALLIATION
FLAG
     PTF5       BIT 22H            ;FLAG OF DEVICE MASKED
     LSC1       BIT 23H            ;1# LOOP SHORT CIRCUIT FLAG
    LSC2    BIT 24H        ;2#.......................
    LSC3    BIT 25H        ;3#.......................
    LSC4    BIT 26H        ;4#.......................
     SATI       BIT 3DH             ;SENSILITY ADJUSTING
     MASB       BIT 3EH              ;TOTAL SWITCH FOR
NON_BLINK
    OPTEST   BIT 43H                 ;LOOP OPEN TEST END FLAG
    PRTF1   BIT 45H        ;PRINT NET CONFIG
    PRTF2   BIT 46H        ;PRINT CONTROL MATRIX
    PRTF3   BIT 47H        ;PRINT DATA LOG
    PRTF4   BIT 48H        ;PRINT LOOP CONFIG
    PRTF5   BIT 49H        ;PRINT EVENT LOG
     LONF       BIT 40H             ;SCAN TABLE
    SCAF       BIT 41H               ;........
    SUCC       BIT 42H
    LOE1       BIT 4AH
    LOE2       BIT 4BH
    FF1        BIT 4CH
    FF2        BIT 4DH
    LPSD    BIT 4EH                  ;0:open loop    1:close loop
    RSTFG      BIT 4FH               ;0: POWER ON RESET      1:
BOOT REST
     FF3        BIT 50H            ;0D1H
;__________________________RS_485 COMMUNICATION
    TABH      EQU 7EH
    FSCF       BIT 30H        ;
    CRCF       BIT 31H         ;CRC
    TSN        BIT 32H         ;
    RSN        BIT 33H         ;
    TFS        BIT 34H         ;
    RFS        BIT 35H         ;
    STNO       BIT 36H         ;0:main point    ,>0:subpoint
    RFG                  BIT 37H;
    CFG     BIT 38H;
    PFG     BIT 39H;SEND TIME
    TRFG       BIT 3AH
    TCFG    BIT 3BH
    TPFG    BIT 3CH
    NETC    BIT 3FH                       ;CHANGE NET CONFIG
    WXYF       BIT 13H         ; BIT13
    JSCF     BIT 14H           ; BIT14
    SCANF     BIT 15H          ; BIT15
    NETNUM          BIT 2DH ; =0 point number>3;    =1 point
number<=3
     CRC1       EQU 3DH
    CRC2    EQU 3EH
    CFCS    EQU 3FH            ;repeat number
    DEST    EQU 4AH            ;source
    NETNO           EQU 4BH           ;web address
    TRCF       EQU 16          ;CFCS
    SPT        EQU    4610H    ;
    TPT        EQU    4611H    ;
    ONLPT EQU    4612H    ;
    RPT        EQU    4613H    ;
    EVPTH EQU    4614H    ;EVENT POINT (H)
    EVPTL  EQU    4615H    ;EVENT POINT (L)
    EVDTAB          EQU    47DFH
    NETCFIG EQU    47E0H
    NETABLE EQU    0A800H           ;web config
     MRPTW      EQU    46E0H
    MRPTR EQU    46E1H
    MSPTW EQU    46E2H
    MSPTR  EQU    46E3H
    SRPTW  EQU    46E4H
    SRPTR  EQU    46E5H
    CF0        BIT        27H
    CF1        BIT        28H
    CEX0    BIT        29H
    CEX1       BIT          2AH
;____________________________________RS_232
COMMUNICATION
     CACS       EQU          0C0H       ;8250 port add.
    CRTB    EQU      28H
    CDLL    EQU          28H
    CDLH    EQU          29H
    CIER    EQU          29H
    CIIR    EQU          2AH
    CLCR    EQU          2BH
    CMCR    EQU          2CH
    CLSR    EQU          2DH
    CMSR    EQU          2EH
    C16M    EQU      30H
    C1S     EQU      31H
    CSPG       EQU          32H
    CSWD    EQU      33H
    CSWP    EQU      34H
    CDAT    EQU      35H
    CTCM    EQU      36H
    CNO1    EQU      37H
    CNO2    EQU      38H
    CNO3    EQU      39H
    CNO4    EQU      3AH
    CDPL    EQU      3BH
    CDPH    EQU      3CH
    CYAS    EQU      50H
    CYAM    EQU      08H
    CMAS    EQU      0FEH
    CBLA    EQU      0D010H
    CHIZ    EQU      10H
```

```
        CBAS   EQU    04H
        CSAD   EQU    0A0H
;___________PRINT_______
        CPF    BIT    2BH     ;print flag
        PT5    BIT    2CH
         PNUM      EQU       32H       ;0_6
         CPBF      EQU       4C00H     ;4C00___4DFF PRINT BUFFER
        SJSP   EQU    4620H
        MDSP   EQU    4630H
        XSJP   EQU    4632H
        SJGS   EQU    4634H
        YSJP   EQU    4636H
         QJQH      EQU       4638H     ;4638__463BH
        SSJP   EQU    463CH   ;REAL TIME PRINT PINT (H L)
        WBAD  EQU    0A800H
        WMAD EQU    0A820H
        WCAD  EQU    0A840H

;_____________________________________________
CBRN   EQU    65H     ;BAUT RATE IDENTIFL EEROR
CBOK   EQU    65H     ;BAUT RATE IDENTIFL OK
CRE1   EQU    65H     ;BAUT RATE IDENTIFL OVERTIEM
CRE2   EQU    65H     ;RECEV DATA OVERTIEM
CRE3   EQU    65H     ;COMMUNICATION LINE EEROR
CEED   EQU    55H     ;EEPROM WRITE EEROR
CEWP   EQU    1901H   ;EEPROM WRITE REPEL
NREC   EQU      1910H         ;RECORD NOT FOUND
CEWO   EQU    21H     ;OPEN EEPROM WRITE REPEL
CTED   EQU    5BH     ;8250 SIO TEST EEROR
CPRR   EQU    68H     ;PRINT NOT CONJUNCTION
;_____________________________________________

;_____RAM Lable Address Table & Variable Name OF LOOP CARD
INTERUPT
MCFA   EQU     2000H  ;first addr. of momentary counter
DPBB   EQU     2100H          ;first addr. of devics data process block
LX_N   EQU     4700H  ;actual numbers of loop cards
FCR1   EQU     4701H          ;orignal value of fire confirm re_scan
FCR2   EQU     4702H          ;orignal value of 2_fault confirm re_scan
TH01   EQU     4703H          ;threshold of 1_fault confirm re_scan
TH02   EQU     4704H          ;orignal value of 1_fault confirm timer
TH03   EQU     4705H          ;orignal value of nr_error confirm timer
TH04   EQU     4706H  ;orignal value of alarm restot. timer
TRE1   EQU     4707H  ;one of 'or' logic for type fault confirm
THTR   EQU     4708H  ;threshold of momentary alarms exceeded
THFI   EQU     4709H  ;threshold of fire confirming delay
THFA   EQU     470AH          ;threshold of pre_fire & 2_fault delay
INTE   EQU     470BH  ;fire intellgent process flag byte(low 3 bits)
THSE   EQU     470CH          ;1_STAGE sensilities adjusting value (3%)
;              470DH          ;2_STAGE.....(6%)
;              470EH          ;3_STAGE.....(9%)
COU1   EQU     470FH          ;counter for confired contaminatin fault
CDM1   EQU     4711H          ;mask byte of devives non_blinking
CDM2   EQU     4712H  ;content of byte 0 in e2prom config.
FAPF   EQU     4713H          ;FIRE THRESHLD OF BEING SCANNED
;              4714H          ;PRE_ALARM THRESHOLD OF BEING
SCANNED
STH1   EQU     4715H  ;BEGIN HOUR TIME OF SENSILITY ADJUSTING
;STS1   EQU     4716H          ;.....MINUTE TIME.....
STH2   EQU     4717H          ;END HOUR TIME OF SENSILITY
ADJUSTING
;STH3   EQU     4718H          ;.....MINUTE TIME.....
DLOG   EQU     4719H  ;NUMBERS OF SENSORS IN DATA LOG
DLEN   EQU     471AH          ;HIGH_8 BYTE OF TOTAL DATA LOG
LENGTH
;              471BH          ;LOW_8 BYTE.....
LOTI   EQU     471CH  ;LOOP 1 FAULT TIMER
;              ____471FH      ;_____LOOP 4 ....
LOSU   EQU     4720H          ;LOOP 1 SCAN STATUS
;              ___4723H       ;____LOOP 4.....
LOEN   EQU     4724H  ;LOOP 1 SCAN END FLAG
;              ____4727H      ;____LOOP 4.....
;LOPE   EQU     4728H  ;LOOP 1 OPEN FLAG
;              __472BH        ;___LOOP 4.....
DALY1   EQU    472CH          ;LOOP CARD 1 RELAY COMMAND
DALY2   EQU    472DH          ;.....2.....
BVCO    EQU    472EH          ;BASE_VALUE COUNTER

DPCT   EQU 4730H       ;DISP MONITOR COUNTER
LX_S   EQU     4740H  ;status data of scaning device
LX_A   EQU     4741H  ;analogue data of scaning device
LX01   EQU     4742H  ;control interface of loop card
LX02   EQU     4744H  ;data interface of loop card
LXSC   EQU     4746H  ;counter of loop card scaning order
LSF1   EQU     4747H          ;flag byte of normal & non_normal scanning
PSPP   EQU     474AH  ;high 8 of pori. scan link
NPSP   EQU     474CH          ;high 8 of non_pori. scan link
FPSP   EQU     474EH          ;first device to be priority_scaned
;              474FH          ;.....non_priority scaned
DM02   EQU     4750H  ;delay initional value of device timer
DM03   EQU     4751H
DM04   EQU     4752H
DM05   EQU     4753H
DPBP   EQU     4754H          ;high_8 pointer of device data process block
LOFA   EQU     4508H          ;loop    1 scan fault byte
LSHR   EQU     4510H  ;loop short circuit fault flag byte
LOSF   EQU     4511H          ;1# LOOP SHORT FAULT COONFIRMED
FLAG
;              4512H      ;2#.....
;              4513H        ;3#.....
;              4514H        ;4#.....
ENDB   EQU     4515H          ;LOOP END_B LOOP_TEST
FLAG:XXXX,D4,D3,D2,D1
RLCF   EQU     455CH          ;flag & command of loop relays control
CQPP   EQU     457FH  ;flag and pointer of queue command to be sent
NFFP   EQU     4600H          ;low 8 pointer of new fire,pre_alarm,fault event
GETC   EQU     4902H  ;low 8 bits of general timer and counter
LOSB   EQU     4A80H  ;1# loop card scan status block (32 bytes)
DSST   EQU     5800H  ;first addr. of device status config. in RAM
LXD1   EQU     0F6H   ;select p2 port of loop card when read from it
LXD2   EQU     0FEH   ;select 646 port of loop card when read from it
LXCH   EQU     0C0H   ;high 8 address of 1# loop card control interface
LXCL   EQU     04H            ;low   8 .....
LXDH   EQU     0C0H   ;high 8 address of 1# loop card data interface
LXDL   EQU     00H            ;low   8 .....
LSBH   EQU     4AH            ;first address high 8 of 1# loop card status
block
LSBL   EQU     80H            ;.....low   8 .....
SDPL   EQU     08H            ;low   8 .....
CQPH   EQU     45H            ;high 8 addr. of queue command FIFO RAM
E2PG   EQU     04H            ;04=base page of e2prom write_protected
SDPH   EQU     4BH            ;high 8 .....
LOFH   EQU     45H            ;high 8 address of saving loop fault
FDLI   EQU     20             ;fire intellgent process delay seconds (20 s)
THCT   EQU     05H            ;devices 'contamination' fault threshold
adjustation
MPPP   EQU     03H            ;PRIOR. SCANNING CONSTANT
FRWP   EQU     4602H  ;
E2SC   EQU     0D010H ;address of control interface storbe to e2prom
DLGP   EQU     8000H  ;FIRST POINTER OF SENSORS DATA LOG
BLOCK
RSII   EQU     0C02AH  ;8250
POWS   EQU     0C050H;
MAST   EQU     0A804H;TOTAL MASK BYTE FOR ALL DEVICES
LOCC   EQU     50             ;LOOP CHECK TIMER VALUE 20S
LOCD   EQU     50             ;LOOP BOARD MONITOR TIME
;=============================================
========================
;          Main Prgram For JB_QB_04_6800 Seris Is As Follows:                ;
;=============================================
========================
        ORG 0000H
        LJMP MAIN
        ORG 0003H
        LJMP IT0N      ;INT0
        ORG 000BH
        LJMP TIMINT    ;T0
        ORG 0013H
         LJMP IT1N          ;INT1 (loop 1,2 ,RS_232,CPU)
        ORG 0017H
WRITE: LJMP WRITE2
        ORG 001BH
        RETI
        ORG 0023H
         LJMP SIOT          ;SIO RS_485 INT
        ORG 0027H
DISP:   LJMP SMB       ;
        ORG 002BH
        RETI           ;T2
        ORG 0030H
;_____________________________
MAIN:     MOV A,SP             ;(SP)__>A
        MOV SP,#SPSP  ;main
        PUSH ACC
        CLR P1.4
         CLR FF3
        MOV R6,#3
T000:   LCALL CDISP    ;reset display card
        JZ TT03
        CPL P1.5
        DJNZ R6,T000
         SETB FF3
TT03:   CPL P1.5
        MOV R0,#SPSP
        CLR A
RMC1:   MOV @R0,A
        DJNZ R0,RMC1
        LCALL RMCL     ;RAM Clear
        CPL P1.5
        MOV PSTUS,#1EH      ;POWER SATUS
        LCALL NETINI ;NET INITIAL
        MOV 41H,#2AH;RESET CODE
        LCALL WRITE2
        LCALL HJINI
        MOV TMOD,#21H
        MOV SCON,#0C0H      ;1100 0000
        MOV TL0,#TML0
        MOV TH0,#TMH0
        MOV 1AH,#30H;SECOND INI VALUE
        MOV 1BH,#0AH        ;10 SECONDS
         LCALL     CINT     ;INITIAL RS_232C   8250
        CPL P1.5
         JNB FF3,TT06
        MOV 41H,#51H ;DISP CARD CPU FAULT CODE
        MOV 43H,#85H ;1000 0101
        LCALL WRITE2
```

```
TT06:  MOV R7,#200
       MOV DPTR,#0E009H
HPHP:  MOVX A,@DPTR
       XRL A,#66H
       JZ HPC
       CPL P1.5
       LCALL HJS009
       DJNZ R7,HPHP
HPC:   LCALL RMCK
       JNB F0,TT04
       MOV 41H,#56H ;RAM CHECK ERROR CODE
       MOV 43H,#84H ;1000 0100
       LCALL WRITE2
       LCALL SMB
TT04:  LCALL DRMCK ;TWO PORTS RAM CHECK
       JNB F0,TT20
       MOV 41H,#57H ;TWO PRTS RAM CHECK ERROR CODE
       MOV 43H,#85H ;1000 0101 DISP BOARD
       LCALL WRITE2
       LCALL SMB
TT20:  LCALL EEPROMCK    ;EEPROM CHECK
       LCALL EPMCK
       JNB F0,TT05
       MOV 41H,#54H ;EPROM CHECK ERROR CODE
       MOV 43H,#84H ;1000 0100
TT05:   MOV R0,#04H              ;to check loop card installiation
       MOV R1,#00H
       LCALL LPINI
       CPL P1.5
       JB F0,RES2
       SETB PTF4          ;N.B:loop card 1 hasn't installed!(PTF4=1)
       MOV 41H,#61H       ;LOOP BOARD NO RESPNSE
       MOV 43H,#86H       ;1000 0110
       LCALL WRITE2
       LCALL SMB
       SJMP RES3
RES2:  MOV DPTR,#LX_N
       MOV A,#1
       MOVX @DPTR,A
RES3:  MOV R0,#05H
       MOV R1,#01H
       LCALL LPINI
       CPL P1.5
       JNB F0,LA01
       MOV DPTR,#LX_N
       MOV A,#2
       MOVX @DPTR,A
       SJMP LA11
LA01:  MOV 41H,#61H       ;LOOP BOARD NO RESPNSE
       MOV 43H,#87H       ;1000 0111
       LCALL WRITE2
       LCALL SMB
LA11:  LCALL TXSCAN
       CPL P1.5
       LCALL STAR         ;FIRST SCANNING
LA04:  SETB TR0
       SETB TR1
       SETB SM2
       SETB REN
       SETB IT0
       CLR IT1
       MOV IP,#10H        ;0001 0011
       MOV IE,#03H        ;0001 0011
       SETB EA
       nop
       nop
       nop
       ; LCALL CTES       ;Test 8250
        CLR C232E
       JZ TT07
        SETB C232E
       MOV 41H,#5BH ;TEST ERROR
       MOV 43H,#84H ;MAIN BOARD
       ;LCALL WRITE2
       LCALL SMB
TT07:   CPL P1.5
       MOV A,#5AH
       MOV WDCON,A
       CLR A
       MOV T3,A           ;INTERNAL   WATCH DOG
       SETB EX1
       CPL P1.5
       MOV DPTR,#LX_N
       MOVX A,@DPTR
       JNZ MIS0
       AJMP ML00A
MIS0:   JB RSTFG,LL72     ;1=RESET
       JB PTF4,LDF0
       MOV A,#0
       MOV B,#0DAH        ;DAH=SWITCH OFF LOOP CARD 1
       LCALL LOPEN1              ;TURN ON LOOP CARD 1 OPEN
TEST
LDF0:  MOV DPTR,#LX_N
       MOVX A,@DPTR
       DEC A
       CPL P1.5
       JZ LD70
       MOV A,#1
       MOV B,#0DAH
       LCALL LOPEN1              ;TURN ON LOOP CARD 2 OPEN
TEST
LD70:   MOV DPTR,#4905H
        MOV A,#03H               ;LOOP_CHECK STARTING TIME
       MOVX @DPTR,A
       SETB PTF3
LD81:  CPL P1.5
       LCALL MB0000
       JNB BIT01,LD81     ;3 SECNDS EXCEEDS
       LCALL HJ0000
       CPL P1.5
       MOV A,#0
       MOV B,#0D0H        ;SWITCH ON LOOP CARD END_A
       LCALL LOPEN1
       MOV DPTR,#LX_N
       MOVX A,@DPTR
       DEC A
       CPL P1.5
       JZ LD71
       MOV A,#1
       MOV B,#0D0H
       LCALL LOPEN1
LD71:   MOV DPTR,#RLCF
LL70:  MOVX A,@DPTR
       JNZ LL70
       INC DPTR
LL71:  MOVX A,@DPTR
       JNZ LL71
LL72:  CLR PTF3
       CLR EX1
       MOV DPTR,#LSBD
       MOV A,#10          ;10 S
       JNB RSTFG,LL73
       MOV A,#3           ;3 S
LL73:  MOVX @DPTR,A               ;START SCAN DELAY TIMER
       CLR BIT09
LDP1:  CPL P1.5
       JNB BIT09,LDP1
       MOV A,#05H
       MOVX @DPTR,A
       CLR BIT09
       CLR PTF3
       SETB EX1
LDP2:  CPL P1.5
       LCALL MB0000
       JNB PTF3,LDP2
       JB PTF4,MIS1
       MOV DPTR,#BUFFER11
       MOV A,#LOCD        ;LOOP CARD 1 MONITOR TIMER
       MOVX @DPTR,A
       CLR BIT10
       MOV DPTR,#4500H
       MOV A,#5
       MOVX @DPTR,A
       LCALL CCTRL        ;SWITCH OFF CONTROL MODUL
MIS1:  MOV DPTR,#LX_N
       MOVX A,@DPTR
       DEC A
       CPL P1.5
       JZ ML0
       MOV DPTR,#BUFFER11
       INC DPTR
       MOV A,#LOCD        ;LOOP CARD 2 MONITOR TIMER
       MOVX @DPTR,A
       CLR BIT11
       MOV DPTR,#4500H
       MOV A,#7
       MOVX @DPTR,A
       LCALL CCTRL
ML0:   MOV DPTR,#4508H
       MOV A,#01H
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       MOV DPTR,#BUFFER12
       MOV A,#TTYL               ;TYPE CHECK   10 MINUTE
       MOVX @DPTR,A
       INC DPTR
       MOV A,#TTYH
       MOVX @DPTR,A
       MOV DPTR,#4905H
        MOV A,#LOCC              ;LOOP_CHECK TIME
       MOVX @DPTR,A
       SETB OPTEST        ;SET LOOP OPEN TEST FLAG
       MOV DPTR,#LOEN
       CLR A
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
       INC DPTR
       MOVX @DPTR,A
```

```
ML00A: MOV DPTR,#DSPT
       MOV A,#5
       MOVX @DPTR,A
       CLR BIT12
       MOV DPTR,#4916H
       MOV A,#255
       MOVX @DPTR,A
       CLR END_BF
       SETB ES                    ;SIO INT
       POP ACC
       JB STNO,ML00        ;START 485 SCAN
       CJNE A,#07H,CODRST  ;CONDITION RESET
       SETB RFG            ;RESET
CODRST:    LCALL DLY100
;_____________________________MAIN CYCLE PROGRAMM
ML00:  CPL P1.5            ;
       ACALL PODET
ML01:  JNB BIT10,ML02
       MOV DPTR,#4A98H
       MOVX A,@DPTR
       CJNE A,#05H,ML010
       CLR BIT10
       SJMP ML02
ML010: MOV R0,#04H
       LCALL LPINI
       MOV DPTR,#DALY1         ;ZJQ COMMAND
       MOVX A,@DPTR
       MOV DPTR,#RLCF
       MOVX @DPTR,A
       MOV DPTR,#BUFFER11
       MOV A,#LOCD
       MOVX @DPTR,A
       CLR BIT10
ML02:  JNB BIT11,ML03
       MOV DPTR,#4AB8H
       MOVX A,@DPTR
       CJNE A,#05H,ML011
       CLR BIT11
       SJMP ML03
ML011: MOV R0,#05H
       LCALL LPINI
       MOV DPTR,#DALY2     ;ZJQ COMMAND
       MOVX A,@DPTR
       MOV DPTR,#RLCF
       INC DPTR
       MOVX @DPTR,A
       MOV DPTR,#BUFFER11
       INC DPL
       MOV A,#LOCD
       MOVX @DPTR,A
       CLR BIT11
ML03:  JNB BIT12,ML04
       CLR BIT12
       CPL P1.5
       MOV 41H,#51H        ;DISP CARD CPU FAULT CODE
       MOV 43H,#85H        ;1000 0101
       LCALL WRITE2
       LCALL CDISP         ;RESET DISPLAY CARD
       CPL P1.5
ML04:  JNB CEX0,ML05
       CPL P1.5
       LCALL RAMCL
       LCALL TXSCAN        ;RESET SCAN TABLE
       CPL P1.5
       CLR CEX0
ML05:  JB RFS,ML05H
       LCALL HJ0000
       CPL P1.5
       JB RFS,ML05H
       LCALL MB0000
       CPL P1.5
ML05H: LCALL TASK485
       CPL P1.5
       LCALL DATALOG
       CPL P1.5
       JNB CEX1,ML06
       MOV DPTR,#LX_N
       MOVX A,@DPTR
       JZ ML06
       MOV DPTR,#4A98H
       MOV A,#01H
       MOVX @DPTR,A
LDF2:  MOV A,#1
       MOV DPTR,#4AB8H
       MOVX @DPTR,A
       LCALL TXSCAN            ;RESET SCAN TABLE
       CLR CEX1
ML06:   LCALL LSHORT           ;SHORT_CIRCUIT TEST
       JNB BIT1A,ML08
       ACALL TIMFLG            ;Y M D 0:0:1      EVENT
RECORD
       CLR BIT1A
       SETB BIT03          ;AUTO TEST FLAG
ML08:  JNB NETC,ML09
       ACALL NETINI
       CLR NETC
       JB STNO,ML09
       LCALL DLY100
ML09:  JB CPF,ML0A
       ACALL DYCLS         ;PRINT HANDLE
ML0A:  JNB 44H,ML0B
       MOV A,C1S
       CJNE A,#10,ML0B
       CLR 44H
       MOV DPTR,#0D010H    ;CLR EEPROM WRITE
       CLR A
       MOVX @DPTR,A
       CPL A
       MOVX @DPTR,A
       CPL A
       MOVX @DPTR,A
       MOV 41H,#65H        ;RS232 COMMUNICATION ERROR
       LCALL WRITE2
       LCALL SMB
ML0B:   JNB E2EOR,ML0C
       MOV 41H,#55H        ;E2 write error
       MOV 43H,#84H        ;1000 0100
       LCALL WRITE2
       LCALL SMB
       CLR E2EOR
ML0C:   JNB END_BF,ML0D
       MOV DPTR,#4600H         ;EVENT SET POINT(LO)
        MOVX A,@DPTR
        MOV R7,A
        INC DPL                ;EVENT EXECUTE POINT(LO)
        MOVX A,@DPTR
        CJNE A,07H,ML0D
       MOV DPTR,#4565H     ;PART/ALL LOSE 1st FLAG
       MOVX A,@DPTR
       JNZ ML0D
       CLR END_BF
       JNB LPSD,ML0D
       ACALL ENDBSUB
ML0D:  AJMP ML00                   ;RETURN MAIN LOOP BEGIN
PBAUT: DW     0000H
       DW     0180H,00C0H
       DW     0060H,0030H
       DW     0018H,000CH
;
PRNN:  DW     0D0AH
DATE:    DW   2020H,0C4EAH
         DW   2020H,0D4C2H
         DW   2020H,0C8D5H
         DW   0CAC2H,0BCFEH
         DW   0BCC7H,0C2BCH
         DW   0A1C3H,2020H
         DW   1AFFH
         DW   0D0AH
TIME:    DW   2020H,0CAB1H
         DW   2020H,0B7D6H
         DW   2020H,2020H
         DW   0BAC5H,0BBFAH
         DW   1AFFH
         DW   0D0AH
SNSE:    DW   2020H,2020H
         DW   0BBD8H,0C2B7H
         DW   2020H,2020H
         DW   0B2E3H,2020H
         DW   2020H,0BAC5H
         DW   0B5D8H,0D6B7H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   0C5A8H,0B6C8H
         DW   3A20H,2020H
         DW   2025H
         DW   1AFFH
         DW   0D0AH
LOOP:    DW   2020H,2020H
         DW   0BBD8H,0C2B7H
         DW   1AFFH
         DW   0D0AH
BRAN:    DW   2020H,2020H
         DW   0B2E3H
         DW   1AFFH
         DW   0D0AH
BATB:  DB     00,06,12,20
       DB     32,40,48,58
       DB     68,76,84,92
       DB     00,00,00,00
QJTB:  DB     00,12,24,36
       DB     50,60,70,80
       DB     88,100,110,120
       DB     130,140,150,160
CQJTB: DB     00,12,24,170
        DB      50,60,70,178
       DB     88,100,110,120
       DB     130,140,150,160
BAND:    DW   2020H,2020H
;        DW   0B2BBH,0D3C3H
         DW   1AFFH
         DW   0B4D3H,0BBFAH
         DW   1AFFH
         DW   0B8B4H,0CABEH
         DW   0C6F7H
```

```
        DW   1AFFH
        DW   5253H,2D32H
        DW   3332H,0BDD3H
        DW   0BFDAH
        DW   1AFFH
        DW   0D6F7H,0BBFAH
        DW   0B0E5H
        DW   1AFFH
        DW   0CFD4H,0CABEH
        DW   0B0E5H
        DW   1AFFH
        DW   0BBD8H,0C2B7H
        DW   0B0E5H,0A2F1H
        DW   1AFFH
        DW   0BBD8H,0C2B7H
        DW   0B0E5H,0A2F2H
        DW   1AFFH
        DW   0B5E7H,0D4B4H
        DW   0B0E5H
        DW   1AFFH
        DW   0CDA8H,0D1B8H
        DW   0B0E5H
        DW   1AFFH
        DW   0D3EFH,0D2F4H
        DW   0B0E5H
        DW   1AFFH
        DW   0BCCCH,0B5E7H
        DW   0C6F7H,0B0E5H
        DW   1AFFH
TYPE:   DW   0B8D0H,0CEC2H          ;0
        DW   0CCBDH,0B2E2H
        DW   0C6F7H
        DW   1AFFH
        DW   0C0EBH,0D7D3H          ;12
        DW   0CCBDH,0B2E2H
        DW   0C6F7H
        DW   1AFFH
        DW   0B9E2H,0B5E7H          ;24
        DW   0CCBDH,0B2E2H
        DW   0C6F7H
        DW   1AFFH
        DW   0BFC9H,0B1E0H          ;36
        DW   0B3CCH,0BCCCH
        DW   0B5E7H,0C6F7H
        DW   1AFFH
        DW   0BCE0H,0CABEH          ;50
        DW   0C4A3H,0BFE9H
        DW   1AFFH
        DW   0BFD8H,0D6C6H          ;60
        DW   0C4A3H,0BFE9H
        DW   1AFFH
        DW   0CAD6H,0B6AFH          ;70
        DW   0B0B4H,0C5A4H
        DW   1AFFH
;       DW   0B2BBH,0B4E6H          ;80
;       DW   0D4DAH
        DW   2020H,2020H,2020H
        DW   1AFFH
        DW   0B8B4H,0CEBBH          ;88
        DW   0BCFCH,0B0B4H
        DW   0CFC2H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;100
        DW   0BCFCH,2031H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;110
        DW   0BCFCH,2032H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;120
        DW   0BCFCH,2033H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;130
        DW   0BCFCH,2034H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;140
        DW   0BCFCH,2035H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;150
        DW   0BCFCH,2036H
        DW   1AFFH
        DW   0B9A6H,0C4DCH          ;160
        DW   0BCFCH,2037H
        DW   1AFFH
        DW   0CCBDH,0B2E2H          ;170
        DW   0C6F7H
        DW   1AFFH
        DW   0C4A3H,0BFE9H          ;178
        DW   1AFFH
SJTB:   DW      0,0,6,12            ;00__0F
        DW    18,24,30,44
        DW    56,66,82,86
        DW    96,106,120,130
;
        DW     140,152,162,172      ;10__1F
        DW    182,192,208,214
        DW    230,244,254,268
        DW    0,286,300,314
;
        DW     314,324,328,338      ;20__2F
        DW    352,362,382,394
        DW    410,420,434,444
        DW    460,466,472,0
;
        DW      0,500,510,520       ;30__3F
        DW    530,540,556,564
        DW    574,580,588,594
        DW    604,0,0,0
;
        DW      0,616,626,636       ;40__4F
        DW    650,1188,656,672
        DW    1196,0,0,0
        DW    0,0,0,0
;
        DW      0,690,702,714       ;50__5F
        DW    724,744,766,782
        DW    804,826,842,1172
        DW    0,0,0,0
;
        DW      0,862,878,900       ;60__6F
        DW    930,962,994,1006
        DW    1016,1032,1042,1058
        DW    1102,1122,1132,1152
;
PMGZ:   DW   0BFAAH,0BBFAH
        DW   1AFFH
        DW   0D5FDH,0B3A3H
        DW   1AFFH
        DW   0BBF0H,0BEAFH
        DW   1AFFH
        DW   0D4A4H,0BEAFH
        DW   1AFFH
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0CAE4H,0C8EBH
        DW   0B5B1H,0C7B0H
        DW   0C3DCH,0C2EBH
        DW   1AFFH
        DW   0C5E4H,0D6C3H
        DW   0D6F7H,0B2CBH
        DW   0B5A5H
        DW   1AFFH
        DW   0CDF8H,0C2E7H
        DW   0C5E4H,0D6C3H
        DW   1AFFH
        DW   0BDA8H,0C1A2H
        DW   0D3EBH,0C9BEH
        DW   0B3FDH,0B2CBH
        DW   0B5A5H
        DW   1AFFH
        DW   0BFD5H
        DW   1AFFH
        DW   0D0DEH,0B8C4H
        DW   0B2CBH,0B5A5H
        DW   1AFFH
        DW   0BBF0H,0BEAFH
        DW   0BBD6H,0B8B4H
        DW   1AFFH
        DW   0B5D8H,0C0EDH
        DW   0CEBBH,0D6C3H
        DW   0C3E8H,0CAF6H
        DW   1AFFH
        DW   0B9CAH,0D5CFH
        DW   0BBD6H,0B8B4H
        DW   1AFFH
        DW   0B8C4H,0B1E4H
        DW   0C3DCH,0C2EBH
        DW   1AFFH
        DW   0B2E2H,0CAD4H
        DW   0D6F7H,0B2CBH
        DW   0B5A5H
        DW   1AFFH
        DW   0B2BDH,0D0D0H
        DW   0B2E2H,0CAD4H
        DW   1AFFH
        DW   0D5FBH,0BBFAH
        DW   0B2E2H,0CAD4H
        DW   1AFFH
        DW   0CAFDH,0BEDDH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0B6AFH,0CCACH
        DW   0B8FAH,0D7D9H
        DW   1AFFH
        DW   0BFD8H,0D6C6H
        DW   0B2D9H,0D7F7H
        DW   0D6F7H,0B2CBH
        DW   0B5A5H
        DW   1AFFH
        DW   0CAB1H,0BCE4H
        DW   1AFFH
        DW   0CFD4H,0CABEH
        DW   0B4F2H,0D3A1H
        DW   0D6F7H,0B2CBH
        DW   0B5A5H
        DW   1AFFH
        DW   0BBD8H,0C2B7H
```

```
        DW   0C5E4H,0D6C3H
        DW   0B4F2H,0D3A1H
        DW   1AFFH
        DW   0B7B5H,0BBD8H
        DW   0D0C5H,0CFA2H
        DW   1AFFH
        DW   0B4F2H,0D3A1H
        DW   0CAFDH,0BEDDH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0CFD4H,0CABEH
        DW   0B4F2H,0D3A1H
        DW   0CAC2H,0BCFEH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0BBD8H,0C2B7H
        DW   0B2CEH,0CAFDH
        DW   0D0DEH,0B8C4H
        DW   1AFFH
        DW   0BFD8H,0D6C6H
        DW   0BED8H,0D5F3H
        DW   0BDA8H,0C1A2H
        DW   1AFFH
        DW   0C3DCH,0C2EBH
        DW   0D6D8H,0CAE4H
        DW   1AFFH
        DW   0BFD5H
        DW   1AFFH
        DW   3144H,0D4A4H
        DW   0CAE4H,0C8EBH
        DW   1AFFH
        DW   0BFD8H,0D6C6H
        DW   0BED8H,0D5F3H
        DW   0D0DEH,0B8C4H
        DW   1AFFH
        DW   0CAC2H,0BCFEH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0B2E9H,0D5D2H
        DW   0B5B1H,0C7B0H
        DW   0CDACH,0C0E0H
        DW   0D0CDH,0CAFDH
        DW   0BEDDH
        DW   1AFFH
        DW   0BCCCH,0B5E7H
        DW   0C6F7H,0B6A8H
        DW   0D2E5H
        DW   1AFFH
        DW   0CAD6H,0B6AFH
        DW   0B2D9H,0D7F7H
        DW   0BCFCH,0B6A8H
        DW   0D2E5H
        DW   1AFFH
        DW   0CAD6H,0B6AFH
        DW   0BFD8H,0D6C6H
        DW   1AFFH
        DW   0C1AAH,0B6AFH
        DW   0CAC2H,0BCFEH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0B8B4H,0CEBBH
        DW   0BCC7H,0C2BCH
        DW   1AFFH
        DW   0BBD8H,0C2B7H
        DW   0BDA8H,0C1A2H
        DW   0D3EBH,0C9BEH
        DW   0B3FDH
        DW   1AFFH
        DW   0CAB1H,0BCE4H
        DW   1AFFH
        DW   0CFFBH,0D2F4H
        DW   0D0AH
        DW   1AFFH
        dw    1affh,1affh,1affh,1affh
        dw    1affh,1affh,1affh,1affh
        dw    1affh,1affh,1affh,1affh
        dw    1affh
QJGZ:   DW   0B8BAH,0D4D8H
        DW   0BFAAH,0C2B7H
        DW   1AFFH
        DW   0B8BAH,0D4D8H
        DW   0B6CFH,0C2B7H
        DW   1AFFH
        DW   0B2E2H,0CAD4H
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0C0E0H,0D0CDH
        DW   0B4EDH,0CEF3H
        DW   1AFFH
        DW   0CEDEH,0CFECH
        DW   0D3A6H,0A3AFH
        DW   0B6AAH,0CAA7H
        DW   1AFFH
        DW   0B7A2H,0CBCDH
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0B5D8H,0D6B7H
        DW   0D6D8H,0C2EBH
        DW   1AFFH
        DW   0CEDBH,0C8BEH
        DW   1AFFH
        DW   0CEB4H,0B6AFH
        DW   0D7F7H
        DW   1AFFH
        DW   0C6C1H,0B1CEH
        DW   1AFFH
        DW   0BDE2H,0B3FDH
        DW   0C6C1H,0B1CEH
        DW   1AFFH
        DW   0CEDEH,0CFECH
        DW   0D3A6H,0BBD6H
        DW   0B8B4H
        DW   1AFFH
HLGZ:   DW   0B2BFH,0B7D6H
        DW   0B6AAH,0CAA7H
        DW   1AFFH
        DW   0C8ABH,0B2BFH
        DW   0B6AAH,0CAA7H
        DW   1AFFH
        DW   0CDA8H,0D1B8H
        DW   0C4DCH,0C1A6H
        DW   0BDB5H,0B5CDH
        DW   1AFFH
        DW   0B6CCH,0C2B7H
        DW   1AFFH
;
;
        DW   0B5D8H,0D6B7H
        DW   0B1E0H,0C2EBH
        DW   0CEAAH,2230H
        DW   3022H
        DW   1AFFH
        DW   0D6D5H,0B6CBH
        DW   0BCCCH,0B5E7H
        DW   0C6F7H,0CEF3H
        DW   0B2D9H,0D7F7H
        DW   1AFFH
DBGZ:   DW   0BFB4H,0C3C5H          ;0A3C3H,0A3D0H
        DW   0B9B7H,0B6AFH ;0A3D5H,0B9CAH
        DW   0D7F7H                ;0D5CFH
        DW   1AFFH
        DW   0A3C3H,0A3D0H
        DW   0A3D5H,0B8B4H
        DW   0CEBBH
        DW   1AFFH
        DW   0C8EDH,0BCFEH
        DW   0B4EDH,0CEF3H
        DW   1AFFH
        DW   0A3C5H,0A3D0H
        DW   0A3D2H,0A3CFH
        DW   0A3CDH,0BCECH
        DW   0B2E9H,0B4EDH
        DW   0CEF3H
        DW   1AFFH
        DW   0A3C5H,0A3C5H
        DW   0A3D0H,0A3D2H
        DW   0A3CFH,0A3CDH
        DW   0D0B4H,0C8EBH
        DW   0B4EDH,0CEF3H
        DW   1AFFH
        DW   0A3D2H,0A3C1H
        DW   0A3CDH,0BCECH
        DW   0B2E9H,0B4EDH
        DW   0CEF3H
        DW   1AFFH
        DW   0CBABH,0B6CBH
        DW   0BFDAH,0A3D2H
        DW   0A3C1H,0A3CDH
        DW   0BCECH,0B2E9H
        DW   0B4EDH,0CEF3H
        DW   1AFFH
        DW   0A3C5H,0A3C5H
        DW   0A3D2H,0A3CFH
        DW   0A3CDH,0BCECH
        DW   0B2E9H,0B4EDH
        DW   0CEF3H
        DW   1AFFH
        DW   0A3D2H,0A3C1H
        DW   0A3CDH,0D0B4H
        DW   0C8EBH,0B4EDH
        DW   0CEF3H
        DW   1AFFH
        DW   0CBABH,0B6CBH
        DW   0BFDAH,0A3D2H
        DW   0A3C1H,0A3CDH
        DW   0D0B4H,0C8EBH
        DW   0B4EDH,0CEF3H
        DW   1AFFH
;
;
ZTGZ:   DW   0B0E5H,0CEB4H
        DW   0BDD3H,0A3AFH
        DW   0CEDEH,0CFECH
        DW   0D3A6H
        DW   1AFFH
        DW   0BFC9H,0B1E0H
```

```
        DW   0B3CCH,0BCCCH
        DW   0B5E7H,0C6F7H
        DW   0BCE0H,0CAD3H
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0B4D3H,0BBFAH
        DW   0CDA8H,0D1B8H
        DW   0B9CAH,0D5CFH
        DW   0A3A8H,0A3D2H
        DW   0A3D3H,0A3ADH
        DW   0A3B4H,0A3B8H
        DW   0A3B5H,0A3A9H
        DW   1AFFH
        DW   0B8B4H,0CABEH
        DW   0C6F7H,0CDA8H
        DW   0D1B8H,0B9CAH
        DW   0D5CFH,0A3A8H
        DW   0A3D2H,0A3D3H
        DW   0A3ADH,0A3B4H
        DW   0A3B8H,0A3B5H
        DW   0A3A9H
        DW   1AFFH
        DW   0B4F2H,0D3A1H
        DW   0BBFAH,0CDA8H
        DW   0D1B8H,0B9CAH
        DW   0D5CFH,0A3A8H
        DW   0A3D2H,0A3D3H
        DW   0A3ADH,0A3B2H
        DW   0A3B3H,0A3B2H
        DW   0A3A9H
        DW   1AFFH
        DW   0D3EFH,0D1D4H
        DW   0B0E5H,0CEB4H
        DW   0BDD3H
        DW   1AFFH
        DW   0D2BAH,0BEA7H
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0B4F2H,0D3A1H
        DW   0BBFAH
;
        DW   0CEB4H,0BDD3H                  ;wei jie
;
        DW   2020H
        DW   2020H
        DW   1AFFH
        DW   0D6F7H,0B5E7H
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0B1B8H,0B5E7H
        DW   0A3AFH,0B3E4H
        DW   0B5E7H,0B9CAH
        DW   0D5CFH
        DW   1AFFH
        DW   0C4DAH,0B2BFH;nuei bu dian yuan gu zhang
        DW   0B5E7H,0D4B4H
        DW   0B9CAH,0D5CFH
        DW   0A3A8H
       DW   0B9FDH,0D1B9H,0A3AFH,0C7B7H,0D1B9H
;       DW   0A3ABH
;        DW   0A3B2H,0A3B4H
;        DW   0A3D6H,0A1A2H
;        DW   0A3ABH,0A3B1H
;        DW   0A3B2H,0A3D6H
;        DW   0A1A2H,0A3ABH
;        DW   0A3B5H,0A3D6H
       DW   0A3A9H,1AFFH
       DW    2020H,2020H,2020H,2020H
       DW    2020H,2020H,2020H,2020H
;        DW   0C1AAH,0B6AFH
        DW   0D6F7H,0B5E7H
        DW   0A3B2H,0A3B4H
        DW   0A3D6H,0B5E7H
        DW   0D4B4H,0B9CAH
        DW   0D5CFH
        DW   1AFFH
        DW   0BDD3H,0B5D8H
        DW   0B9CAH,0D5CFH
        DW   1AFFH
        DW   0D6F7H,0B0E5H
        DW   0CFD4H,0CABEH
        DW   0B0E5H,0CDA8H
        DW   0D1B8H,0B9CAH
        DW   0D5CFH
        DW   1AFFH
;
        DW   0BBF0H,0BEAFH,0BCCCH,0B5E7H,0C6F7H
        DW   0BCE0H,0CABEH,0B9CAH,0D5CFH,1AFFH
        ;huo jin ji dian qi jian shi gou zhang
;
        DW   5253H,3233H,3220H,0B2E2H,0CAD4H
        DW   0B9CAH,0D5CFH,1AFFH                ;rs232 ce shi gou
zhang
;
        DW   0CEDEH,0CFECH,0D3A6H,1AFFH         ;wu xiang ying
;
       DW   0B6CFH,0CFDFH,1AFFH                 ;duan xian
;
PITB:   DW    0D0AH
        DW    2020H,2020H
        DW    0BBD8H,0C2B7H
        DW    0C6F7H,0BCFEH
        DW    0C5E4H,0D6C3H
        DW    0B1EDH,0A1C3H
        DW    2020H,2020H
        DW    2020H,2020H
        DW    2020H,2020H
        DW    2020H,2020H
        DW    0D0AH
        DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   0D3C5H,0CFC8H
         DW   2020H,0C8B7H
         DW   0C8CFH,2020H
        DB    20H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,2020H
         DW   2020H,0B7A7H
         DW   2020H,0D6B5H
         DW   0A3A8H,0A3A5H
         DW   0A3A9H,2020H
         DW   2020H,0D1D3H
         DW   0CAB1H,0A3A8H
         DW   0C3EBH,0A3A9H
         DW   0D0AH
         DW   0B5D8H,0D6B7H
         DW   2020H,2020H
         DW   0C0E0H,2020H
         DW   0D0CDH,2020H
         DW   2020H,0C6C1H
         DW   0B1CEH,2020H
         DW   0C9A8H,0C3E8H
         DW   2020H,0B7BDH
         DW   0CABDH,2020H
         DW   0C7F8H,0BAC5H
         DW   2020H,0B2E3H
         DW   0BAC5H,2020H
         DW   0BBF0H,0BEAFH
         DW   2020H,0D4A4H
         DW   0BEAFH,2020H
         DW   0B9CAH,0D5CFH
         DW   2020H,0BBF0H
         DW   0BEAFH,2020H
         DW   0B9CAH,0D5CFH
         DW   2020H,2020H
         DW   0B5D8H,0C0EDH
         DW   0CEBBH,0D6C3H
         DW   0D0AH
         DW   1AFFH
;
WBJT:    DW   0D0AH
        DW   0B1BEH,0BBFAH  ;Ben Ji Pei Zhi Bao
         DW   0C5E4H,0D6C3H
         DW   0B1EDH
         DW   0D0AH
         DW   0BBFAH,0C6F7H
         DW   0BAC5H,2020H
         DW   0CAB5H,0CAB1H
         DW   0B4F2H,0D3A1H
         DW   2020H,0D3EFH
         DW   0D1D4H,0CCE1H
         DW   0CABEH,2020H
         DW   0C8B7H,0C8CFH
         DW   0B5C6H,2020H
         DW   0BBD8H,0C2B7H
         DW   0C1ACH,0BDD3H
         DW   2020H,0C9A8H
         DW   0C3E8H,0BCE4H
         DW   0B8F4H,28C3H
         DW   0EB29H,2020H
         DW   3438H,35BFH
         DW   0DACBH,0D9C2H
         DW   0CA20H,2032H
         DW   3332H,0BFDAH
         DW   0CBD9H,0C2CAH
         DW   2020H,2020H
         DW   2020H,0D5BEH
         DW   0C3E8H,0CAF6H
         DW   0D0AH
        DW   1AFFH
CENB:    DW   0D3D0H,0D0A7H
         DW   1AFFH
CDEN:    DW   0CEDEH,0D0A7H
         DW   1AFFH
CSDE:    DW   0B2BBH,0C9C1H
         DW   1AFFH
CSEN:    DW   0C9C1H,0CBB8H
         DW   1AFFH
CLKP:    DW   0BFAAH,0BBB7H
         DW   1AFFH
CLBP:    DW   0B1D5H,0BBB7H
```

```
            DW   1AFFH
;
            DW   0D3D0H,0C5E4H
            DW   0D6C3H,2020H
            DW   0CEDEH,0C5E4H
            DW   0D6C3H
            DW   0D0AH
           DW   1AFFH
;
WMTB:      DW   0D0AH
           DW   0D6F7H,0CDF8H  ;Zhou wang pei zhi bao
            DW   0C5E4H,0D6C3H
            DW   0B1EDH
            DW   0D0AH
            DW   0BBFAH,0C6F7H
            DW   0BAC5H,2020H
            DW   2020H,0C6C1H
            DW   0B1CEH,2020H
            DW   0CDA8H,0D0C5H
            DW   0B9CAH,0D5CFH
            DW   2020H,0CCF5H
            DW   0BCFEH,0B8B4H
            DW   0CEBBH,0CFFBH
            DW   0D2F4H,2020H
            DW   0CAC2H,0BCFEH
            DW   0B9CAH,0D5CFH
            DW   0D0AH
           DW   1AFFH
;
CPDN:      DW   0CEDEH,0C6C1H
           DW   0B1CEH,1AFFH
CPEN:      DW   0D3D0H,0C6C1H
           DW   0B1CEH,1AFFH
;
CGDN:      DW   0B2BBH,0B8F4H
           DW   0C0EBH,1AFFH
CGEN:      DW   0B8F4H,0C0EBH
           DW   1AFFH
;
CCEN:      DW   0B4ABH,0CBCDH
           DW   1AFFH
CCDN:      DW   0B2BBH,0B4ABH
           DW   0CBCDH,1AFFH
;
WFUT:      DW   0D0AH
           DW   0B4D3H,0CDF8H  ;Fu shi qi pei zhi bao
            DW   0C5E4H,0D6C3H
            DW   0B1EDH
            DW   0D0AH
            DW   0B8B4H,0CABEH
            DW   0C6F7H,0BAC5H
            DW   2020H,2020H
            DW   0C6C1H,0B1CEH
            DW   2020H,0CDA8H
            DW   0D0C5H,0B9CAH
            DW   0D5CFH,2020H
            DW   0BCE0H,0CAD3H
;           DW   0C7F8H
;           DW   28B2H,0E329H
           DW    0B2E3H
            DW   0D0AH
           DW   1AFFH
WFUL:      DW   0C8ABH,0B2BFH
            DW   1AFFH
;
RBTO:  DW         0D0AH
           DW    2020H,2020H
           DW    0BAC5H,0BBFAH
           DW    2020H,2020H
           DW    2020H,2020H
;          DW    0BAC5H,0BBD8H
;          DW    0C2B7H,2020H
           DW    2020H,2020H
           DW    2020H,2020H
           DW    2020H,2020H
           DW    2020H,2020H
           DW    0BAC5H,0B5D8H
           DW    0D6B7H,2020H
           DW    2020H,2020H
           DW    0B2C9H,0D1F9H
           DW    0BCE4H,0B8F4H
           DW    28C3H,0EB29H
           DW    2020H,2020H
           DW    2020H,2020H
           DW    2020H,2020H
           DW    2020H,2020H
;          DW    2020H,2020H
           DW    2020H
PTIME: DW    2020H,0C4EAH
           DW    2020H,0D4C2H
           DW    2020H,0C8D5H
           DW    2020H,0CAB1H
           DW    2020H,0B7D6H
           DW    2020H,0C3EBH
           DW    1AFFH
;
CTRL:      DW   0D0AH
           DW   0BFD8H,0D6C6H
           DW   0BED8H,0D5F3H
           DW   0C2DFH,0BCADH
           DW   0B9D8H,0CFB5H
           DW   0B1EDH,0A1C3H
           DW   0D0AH
           DW   0D0AH
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   0CAE4H,2020H
           DW   2020H,0C8EBH
           DW   2020H,2020H
           DW   0CFEEH,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   2020H,2020H
           DW   0CAE4H,2020H
           DW   2020H,0B3F6H
           DW   2020H,2020H
           DW   0CFEEH
           DW   0D0AH
;
           DW   0C7F8H,0BAC5H
           DB   20H
           DW   0B2E3H,0BAC5H
           DB   20H
           DW   0BBFAH,0C6F7H,0BAC5H
           DB   20H
;          DW   0BBD8H,0C2B7H,0BAC5H
;          DB   20H
           DW   0B5D8H,0D6B7H
           DB   20H
           DW   0C6F7H,0BCFEH
           DW   0C0E0H,0D0CDH
           DW   2020H,2020H
           DW   0C2DFH,0BCADH
          DW   0B9D8H,0CFB5H
           DW   2020H,2020H
           DW   2020H
;
           DW   0C7F8H,0BAC5H
           DB   20H
           DW   0B2E3H,0BAC5H
           DB   20H
           DW   0BBFAH,0C6F7H,0BAC5H
           DB   20H
;          DW   0BBD8H,0C2B7H,0BAC5H
;          DB   20H
           DW   0B5D8H,0D6B7H
           DB   20H
           DW   0CAE4H,0B3F6H
           DW   0C6F7H,0BCFEH
           DB   20H
           DW   0CAE4H,0B3F6H
           DW   0B7BDH,0CABDH
           DB   20H
;          DW   0CAE4H,0B3F6H
           DW   0D7B4H,0CCACH
           DB   20H
           DW   0D1D3H
           DW   0CAB1H,28C3H
           DW   0EB29H
           DW   0D0AH
           DW   1AFFH
;
TBJP:  DB    3,3,3,0   ;0
           DB    0,0,3,3          ;0,3,3,3
           DB    3,3,3,3
           DB    0,3,0,3
           DB    3,3,3,3   ;1
           DB    3,3,6,3
           DB    3,3,3,3
           DB    0,3,3,3          ;3,3,3,3
           DB    3,3,3,3 ;2
           DB    3,3,3,3
           DB    3,3,3,3
            DB      4,3,3,7          ;2F___AUOT TEST
           DB    0,0,0,0 ;3
           DB    0,0,0,0
           DB    0,0,0,0
           DB    0,0,0,0
           DB    1,1,1,1   ;4
           DB    1,1,1,1
           DB    1,1,1,1
           DB    1,1,1,1
           DB    2,2,2,2   ;5
           DB    2,2,2,2
           DB    2,2,2,2
           DB    2,2,2,2
           DB    1,2,3,3   ;6
           DB    5,3,3,3
```

```
        DB      3,3,3,3
        DB      3,3,3,3
        DB      3,3,3,3   ;7
        DB      3,3,3,3
        DB      3,3,3,3
        DB      3,3,3,3
;
;_____________________________________ RS_232
;
CCCC:   CLR     A
        MOV     C1S,A
        CLR     C
        MOV     A,CSWD
        RLC     A
        MOV     DPTR,#CITB
        JMP     @A+DPTR
        RET
CITB:   AJMP    CIP0            ;(CSWD)=0
        AJMP    CIP1              ;(CSWD)=1
        AJMP    CIP2            ;(CSWD)=2
        AJMP    CIP3            ;(CSWD)=3
        AJMP    CIP4              ;(CSWD)=4
        AJMP    CIP5              ;(CSWD)=5
        AJMP    CIP6              ;(CSWD)=6
        AJMP    CIP7            ;(CSWD)=7
        AJMP    CIP8              ;(CSWD)=8
        AJMP    CIP9              ;(CSWD)=9
        AJMP    CIPA              ;(CSWD)=10
        AJMP    CIPB              ;(CSWD)=11
        AJMP    CIPC              ;(CSWD)=12
        AJMP    CIPD              ;(CSWD)=13
        AJMP    CIPE              ;(CSWD)=14
        AJMP    CIPF              ;(CSWD)=15
        AJMP    CIPH              ;(CSWD)=10H
        AJMP    CIPI            ;(CSWD)=11H
        AJMP    CIPJ            ;(CSWD)=12H
CIP0:   LJMP    CBUT
CIP1:   LJMP    CBOT
CIP2:   LJMP    COMM
CIP3:   LJMP    CNUM
CIP4:   LJMP    CNM1
CIP5:   LJMP    CNM2
CIP6:   LJMP    CNM3
CIP7:   LJMP    CMD1
CIP8:   LJMP    CMD2
CIP9:   LJMP    CMD3
CIPA:   LJMP    CTE1
CIPB:   LJMP    CTE2
CIPC:   LJMP    CTE3
CIPD:   LJMP    CTE4
CIPE:   LJMP    CTE5
CIPF:   LJMP    CTX2
CIPH:   LJMP    CTE6
CIPI:   LJMP    CTE7
CIPJ:   LJMP    CTE8
        RET
;***********************************************
COMM:   LCALL   CRCV
        CJNE    A,#55H,COMD
        MOV     CSWD,#03H
        RET
COMD:   CJNE    A,#7EH,COML
        MOV     CSWD,#07H
        RET
COML:   MOV     CSWD,#02H
        RET
CMD1:   LCALL   CRCV
        ANL     A,#07H
        RL      A
        RL      A
        MOV     CSWP,A
        MOV     CSWD,#08H
        RET
CMD2:   LCALL   CRCV
CMD3:   MOV     DPTR,#COMJ
        MOV     A,CSWP
        JMP     @A+DPTR
COMJ:   LJMP    COML
        NOP
        LJMP    CTX1
        NOP
        LJMP    CTX2
        NOP
        LJMP    CTX3
        NOP
        LJMP    CTX4
        NOP
        LJMP    CTX5
        NOP
        LJMP    CTX6
CTX1:   MOV     CNO2,#00H
        MOV     CNO1,#03H
        MOV     R4,#00H
        CPL P1.5
        DJNZ    R4,$
        DJNZ    R4,$
        CPL P1.5
        DJNZ    R4,$
        DJNZ    R4,$
        CPL P1.5
CTXA:   MOV     CNO4,#00H
        MOV     A,CNO2
        LCALL   CADD
CTEN:   MOV     A,#55H
;       SETB    44H
        LCALL   CTET
        MOV     CSWD,#0AH
        RET
CTE1:   LCALL   CTRD
        MOV     A,CNO2
        LCALL   CTET
        MOV     CSWD,#0BH
        RET
CTE2:   LCALL   CTRD
        MOV     A,CNO3
        LCALL   CTET
        MOV     CSWD,#0CH
        RET
CTE3:   LCALL   CTRD
CTPP:   MOV DPTR,#4500H
        MOV A,CSPG
        MOVX @DPTR,A
        MOV     DPTR,#0D010H
        MOVX    @DPTR,A
        MOV     DPH,CDPH
        MOV     DPL,CDPL
        MOVX    A,@DPTR
        INC     DPTR
        MOV     CDPH,DPH
        MOV     CDPL,DPL
        LCALL   CTET
        MOV     CSWD,#0DH
        RET
CTE4:   LCALL   CTRD
        DJNZ    CNO3,CTPP
        CLR     A
        CPL     A
        LCALL   CTET
        MOV     CSWD,#0EH
        RET
CTE5:   LCALL   CTRD
CTX2:   MOV     A,CNO4
        ANL     A,#0FH
        JZ      CTXC
        DJNZ    CNO1,CTXA
        MOV     CSWD,#02H
        MOV     41H,#CRE3
        LCALL   SMB
        AJMP    CTX5
        RET
CTXC:   MOV     CNO1,#03H
        INC     CNO2
        MOV     R0,CNO2
        CJNE    R0,#110,CTXA
CTE9:   MOV     A,#7EH
        LCALL   CTET
        MOV     CSWD,#10H
        RET
CTE6:   LCALL   CTRD
        MOV     A,#44H
        LCALL   CTET
        MOV     CSWD,#11H
        RET
CTE7:   LCALL   CTRD
        MOV     A,#0FFH
        LCALL   CTET
        MOV     CSWD,#12H
        RET
CTE8:   LCALL   CTRD
        MOV     A,CNO4
        JZ      CTE0
        DJNZ    CNO1,CTE9
        MOV     CSWD,#02H
        MOV     41H,#CRE3
        LCALL   SMB
        AJMP    CTX5
        RET
CTE0:   MOV     CSWD,#02H
        RET
CTX3:   LJMP    COML
CTX4:   MOV     DPTR,#CBLA
        MOV     A,#CHIZ
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
;       CLR     44H
CHECK:  MOV     DPH,#CACS
        MOV     DPL,#CIIR
        MOVX    A,@DPTR
        ANL     A,#06H
        CJNE    A,#06H,NXT1
        LJMP    LINE
NXT1:   CJNE    A,#04H,NXT2
        LJMP    RECE
```

```
NXT2:   CJNE    A,#02H,NXT3
        LJMP    SEND
NXT3:   CJNE    A,#00H,NXT4
        LJMP    MODEM
NXT4:   MOV     DPH,#CACS
        MOV     DPL,#CIIR
        MOVX    A,@DPTR
        ANL     A,#01H
        JZ      CHECK
        MOV     CSWD,#00H
        LCALL   CINT
        RET
LINE:   MOV     DPH,#CACS
        MOV     DPL,#CLSR
        MOVX    A,@DPTR
        LJMP    NXT4
RECE:   NOP
WT3:    MOV     DPH,#CACS
        MOV     DPL,#CLSR
        MOVX    A,@DPTR
        ANL     A,#01H
        JZ      WT3
        MOV     DPH,#CACS
        MOV     DPL,#CRTB
        MOVX    A,@DPTR
        LJMP    NXT4
SEND:   NOP
WT4:    MOV     DPH,#CACS
        MOV     DPL,#CLSR
        MOVX    A,@DPTR
        ANL     A,#20H
        JZ      WT4
        MOV     DPH,#CACS
        MOV     DPL,#CRTB
        MOV     A,#0FFH
        MOVX    @DPTR,A
        LJMP    NXT4
MODEM:  MOV     DPH,#CACS
        MOV     DPL,#CMCR
        MOVX    A,@DPTR
        LJMP    NXT4
CTX5:   SETB    CEX0            ;ON LOOP SCAN
        CLR     44H
        CLR A
        MOV DPTR,#4500H
        MOVX @DPTR,A
        MOV     DPTR,#0D010H
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
        MOV     R7,#20H
CWAT2:  MOV     R6,#00H
CWAT1:  MOV     R5,#00H
        CPL     P1.5
CWAT0:  DJNZ    R5,CWAT0
        DJNZ    R6,CWAT1
        DJNZ    R7,CWAT2
        AJMP CTX4
        RET
CTX6:   MOV DPTR,#4A98H         ;STOP LOOP SCAN INT
        MOV A,#01H
        MOVX @DPTR,A
        MOV DPTR,#4AB8H
        MOVX @DPTR,A
        MOV     CSWD,#02H
        RET
;_____________________
CNUM:   SETB    P1.1
        JNB     P1.1,CENW
        MOV     DPTR,#CEWP      ;Display "EEPROM write repel"
        MOV 41H,DPH
        MOV 42H,DPL
        LCALL   SMB
        MOV     CSWD,#00H
        LJMP    CTX5
        RET
CENW:   CLR     A
        MOV     C1S,A
        SETB    44H
        LCALL   CRCV
        LCALL   CADD
        MOV     CSWD,#04H
        RET
CNM1:   LCALL   CRCV
        MOV     CNO3,A
        MOV     CSWD,#05H
        RET
CNM2:   LCALL       CRCE
        MOV B,A
        MOV DPTR,#4500H
        MOV A,CSPG
        MOVX @DPTR,A
        MOV DPTR,#0D010H
        MOVX @DPTR,A
        MOV A,B
        MOV     DPH,CDPH
        MOV     DPL,CDPL
        MOVX    @DPTR,A
        MOV     A,C16M
        ADD     A,#CYAM
        MOV     R5,A
CNEX:   MOV     A,C16M
        XRL     A,R5
        CPL P1.5
        JZ      CEER
        MOVX    A,@DPTR
        XRL     A,B
        JNZ     CNEX
CCNXT:      INC     DPTR
        MOV     CDPH,DPH
        MOV     CDPL,DPL
        MOV     A,B
        MOV     DPL,#CRTB
        MOV     DPH,#CACS
        MOVX    @DPTR,A
        MOV     CSWD,#06H
        RET
CNM3:   DJNZ    CNO3,CNM2
        LCALL   CRCV
        MOV     CSWD,#02H
        RET
CEER:   SETB    CEX0            ;SETE EEPROM WRITE REPEL AND ON
LOOP SCAN
        CLR     44H
        CLR A
        MOV DPTR,#4500H
        MOVX @DPTR,A
        MOV     DPTR,#0D010H
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
        CPL     A
        MOVX    @DPTR,A
        MOV     41H,#CEED
        LCALL   WRITE2
        LCALL   SMB
        LJMP    CTX5
        RET
CRCV:   NOP
CRRV:   MOV     DPL,#CRTB
        MOV     DPH,#CACS
        MOVX    A,@DPTR
        MOVX    @DPTR,A
        MOV     DPL,#CLSR
        PUSH    ACC
CWAT:   MOVX    A,@DPTR
        JNB     ACC.6,CWAT
        POP     ACC
        RET
CRCE:   NOP
        MOV     DPL,#CRTB
        MOV     DPH,#CACS
        MOVX    A,@DPTR
        RET
CTET:   MOV     DPH,#CACS       ;Emit data
        MOV     DPL,#CRTB
        MOVX    @DPTR,A
        MOV     CDAT,A
        RET
CTRD:   LCALL   CRCE
        XRL     A,CDAT
        JZ      CTNT
        MOV     A,#04H
        ORL     A,CNO4
        MOV     CNO4,A
CTNT:   RET
CADD:   MOV     B,#25           ;Compute black num and add
        DIV     AB
        MOV     C,ACC.0
        MOV     ACC.7,C
        MOV     C,ACC.1
        MOV     ACC.0,C
        MOV     C,ACC.7
        MOV     ACC.1,C
        ANL     A,#07H
        ADD     A,#CBAS
        MOV     CSPG,A
        MOV     A,B
        ADD     A,#CSAD
        MOV     CDPH,A
        MOV     CDPL,#00H
        MOV     A,CSPG
        CJNE    A,#8,CADE
        MOV     A,B
        CJNE    A,#10,CADE
        MOV     CNO3,#48
        RET
CADE:   MOV     CNO3,#00H
        RET
CBUT:   MOV     CNO1,#05H       ;Baut rate identifl
        MOV     CSWD,#01H
CBOT:   MOV     DPH,#CACS
        MOV     DPL,#CLSR
        MOVX    A,@DPTR
```

```
        JB      ACC.0,CBRV
        JB      ACC.3,CBL3
        JB      ACC.4,CBL3
        RET
CBRV:   MOV     DPH,#CACS
        MOV     DPL,#CRTB
        MOVX    A,@DPTR
        XRL     A,#CMAS
        JZ      CBL4
CBL3:   MOV     DPH,#CACS
        MOV     DPL,#CLCR
        MOVX    A,@DPTR
        ORL     A,#80H
        MOVX    @DPTR,A
        MOV     DPL,#CDLL
        MOVX    A,@DPTR
        CLR     C
        RLC     A
        JZ      CBL5
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    A,@DPTR
        RLC     A
        MOVX    @DPTR,A
CBL8:   MOV     DPL,#CLCR
        MOVX    A,@DPTR
        ANL     A,#7FH
        MOVX    @DPTR,A
        RET
CBL5:   MOV     A,#0CH
        MOVX    @DPTR,A
        CLR     A
        INC     DPTR
        MOVX    @DPTR,A
        DJNZ    CNO1,CBL8
        MOV     DPL,#CLCR
        MOVX    A,@DPTR
        ANL     A,#7FH
        MOVX    @DPTR,A
        CLR     A
        MOV     CSWD,A
        RET
CBL4:   MOV     DPH,#CACS
        MOV     DPL,#CRTB
        MOV     A,#CMAS
        MOVX    @DPTR,A
        MOV     DPH,#CACS
        MOV     DPL,#CIER
        MOV     A,#01H
        MOVX    @DPTR,A
        MOV     CSWD,#02H
CBTW:   MOV     DPH,#CACS
        MOV     DPL,#CLCR
        MOVX    A,@DPTR
        ORL     A,#80H
        MOVX    @DPTR,A
        MOV     DPL,#CDLL
        MOVX    A,@DPTR
        MOV     R3,A
        MOV     DPL,#CLCR
        MOVX    A,@DPTR
        ANL     A,#7FH
        MOVX    @DPTR,A
        MOV     A,R3
        CJNE    A,#0CH,CBI1
        MOV     R3,#96H
        LJMP    CBI6
CBI1:   CJNE    A,#18H,CBI2
        MOV     R3,#48H
        LJMP    CBI6
CBI2:   CJNE    A,#30H,CBI3
        MOV     R3,#24H
        LJMP    CBI6
CBI3:   CJNE    A,#60H,CBI4
        MOV     R3,#12H
        LJMP    CBI6
CBI4:   CJNE    A,#0C0H,CBI5
        MOV     R3,#06H
        LJMP    CBI6
CBI5:   MOV     R3,#03H
CBI6:   MOV     A,R3
        RET
;
TAB150: DB 00H
DW 0102H,0305H,0607H,0809H,0A0CH,0D0EH,0F10H,1113H,1415H,1617H
DW 1819H,1B1CH,1D1EH,1F20H,2223H,2425H,2627H,292AH,2B2CH,2D2EH
DW 2F31H,3233H,3435H,3638H,393AH,3B3CH,3D3FH,4041H,4243H,4445H
DW 4748H,494AH,4B4CH,4E4FH,5051H,5253H,5556H,5758H,595AH,5B5DH
DW 5E60H,6162H,6364H,6566H,6768H,696BH,6C6DH,6E6FH,7072H,7374H
DW 7576H,7779H,7A7BH,7C7DH,7E80H,8182H,8384H,8587H,8889H,8A8BH
DW 8C8DH,8F90H,9192H,9394H
        ORG 7E00H
CRCTAB:     DB 00H,00H,89H,11H,12H,23H,9BH,32H
        DB 24H,46H,0ADH,57H,36H,65H,0BFH,74H
        DB 48H,8CH,0C1H,9DH,5AH,0AFH,0D3H,0BEH
        DB 6CH,0CAH,0E5H,0DBH,7EH,0E9H,0F7H,0F8H
        DB 81H,10H,08H,01H,93H,33H,1AH,22H
        DB 0A5H,56H,2CH,47H,0B7H,75H,3EH,64H
        DB 0C9H,9CH,40H,8DH,0DBH,0BFH,52H,0AEH
        DB 0EDH,0DAH,64H,0CBH,0FFH,0F9H,76H,0E8H
        DB 02H,21H,8BH,30H,10H,02H,99H,13H
        DB 26H,67H,0AFH,76H,34H,44H,0BDH,55H
        DB 4AH,0ADH,0C3H,0BCH,58H,8EH,0D1H,9FH
        DB 6EH,0EBH,0E7H,0FAH,7CH,0C8H,0F5H,0D9H
        DB 83H,31H,0AH,20H,91H,12H,18H,03H
        DB 0A7H,77H,2EH,66H,0B5H,54H,3CH,45H
        DB 0CBH,0BDH,42H,0ACH,0D9H,9EH,50H,8FH
        DB 0EFH,0FBH,66H,0EAH,0FDH,0D8H,74H,0C9H
        DB 04H,42H,8DH,53H,16H,61H,9FH,70H
        DB 20H,04H,0A9H,15H,32H,27H,0BBH,36H
        DB 4CH,0CEH,0C5H,0DFH,5EH,0EDH,0D7H,0FCH
        DB 68H,88H,0E1H,99H,7AH,0ABH,0F3H,0BAH
        DB 85H,52H,0CH,43H,97H,71H,1EH,60H
        DB 0A1H,14H,28H,05H,0B3H,37H,3AH,26H
        DB 0CDH,0DEH,44H,0CFH,0DFH,0FDH,56H,0ECH
        DB 0E9H,98H,60H,89H,0FBH,0BBH,72H,0AAH
        DB 06H,63H,8FH,72H,14H,40H,9DH,51H
        DB 22H,25H,0ABH,34H,30H,06H,0B9H,17H
        DB 4EH,0EFH,0C7H,0FEH,5CH,0CCH,0D5H,0DDH
        DB 6AH,0A9H,0E3H,0B8H,78H,8AH,0F1H,9BH
        DB 87H,73H,0EH,62H,95H,50H,1CH,41H
        DB 0A3H,35H,2AH,24H,0B1H,16H,38H,07H
        DB 0CFH,0FFH,46H,0EEH,0DDH,0DCH,54H,0CDH
        DB 0EBH,0B9H,62H,0A8H,0F9H,9AH,70H,8BH
        DB 08H,84H,81H,95H,1AH,0A7H,93H,0B6H
        DB 2CH,0C2H,0A5H,0D3H,3EH,0E1H,0B7H,0F0H
        DB 40H,08H,0C9H,19H,52H,2BH,0DBH,3AH
        DB 64H,4EH,0EDH,5FH,76H,6DH,0FFH,7CH
        DB 89H,94H,00H,85H,9BH,0B7H,12H,0A6H
        DB 0ADH,0D2H,24H,0C3H,0BFH,0F1H,36H,0E0H
        DB 0C1H,18H,48H,09H,0D3H,3BH,5AH,2AH
        DB 0E5H,5EH,64H,4FH,0F7H,7DH,7EH,6CH
        DB 0AH,0A5H,83H,0B4H,18H,86H,91H,97H
        DB 2EH,0E3H,0A7H,72H,3CH,0C0H,0B5H,0D1H
        DB 42H,29H,0CBH,38H,50H,0AH,0D9H,1BH
        DB 66H,6FH,0EFH,7EH,74H,4CH,0FDH,5DH
        DB 8BH,0B5H,02H,0A4H,99H,96H,10H,87H
        DB 0AFH,0F3H,26H,0E2H,0BDH,0D0H,34H,0C1H
        DB 0C3H,39H,4AH,28H,0D1H,1AH,58H,0BH
        DB 0E7H,7FH,6EH,6EH,0F5H,5CH,7CH,4DH
        DB 0CH,0C6H,85H,0D7H,1EH,0E5H,97H,0F4H
        DB 28H,80H,0A1H,91H,3AH,0A3H,0B3H,0B2H
        DB 44H,4AH,0CDH,5BH,56H,69H,0DFH,78H
        DB 60H,0CH,0E9H,1DH,72H,2FH,0FBH,3EH
        DB 8DH,0D6H,04H,0C7H,9FH,0F5H,16H,0E4H
        DB 0A9H,90H,20H,81H,0BBH,0B3H,32H,0A2H
        DB 0C5H,5AH,4CH,4BH,0D7H,79H,5EH,68H
        DB 0E1H,1CH,68H,0DH,0F3H,3FH,7AH,2EH
        DB 0EH,0E7H,87H,0F6H,1CH,0C4H,95H,0D5H
        DB 2AH,0A1H,0A3H,0B0H,38H,82H,0B1H,93H
        DB 46H,6BH,0CFH,7AH,54H,48H,0DDH,59H
        DB 62H,2DH,0EBH,3CH,70H,0EH,0E9H,1FH
        DB 8FH,0F7H,06H,0E6H,9DH,0D4H,14H,0C5H
        DB 0ABH,0B1H,22H,0A0H,0B9H,92H,30H,83H
        DB 0C7H,7BH,4EH,6AH,0D5H,58H,5CH,49H
        DB 0E3H,3DH,6AH,2CH,0F1H,1EH,78H,0FH
;
        END
```

Appendix B: TFM13 AST in XML

```
<Statements>
    <Var>
        <Assigns>
            <Assign>
                <Var_Lvalue value="cc"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
            <Assign>
                <Var_Lvalue value="cc1"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
            <Assign>
                <Var_Lvalue value="destination"></Var_Lvalue>
                <Number value="0"></Number>
            </Assign>
        </Assigns>
        <Statements>
            <A_S>
                <Name value="_enter_"></Name>
                <Actions>
                    <Action>
                        <Name value="_enter_"></Name>
                        <Statements>
                            <Comment value="&lt;ENTRY POINT&gt; "></Comment>
                            <Comment value="&lt;NAME=FMT001A0&gt; "></Comment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r7"></Var_Lvalue>
                                    <Variable value="__r7_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r11"></Var_Lvalue>
                                    <Variable value="__r11_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r12"></Var_Lvalue>
                                    <Variable value="__r12_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r13"></Var_Lvalue>
                                    <Variable value="__r13_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="r14"></Var_Lvalue>
                                    <Variable value="__r14_init__"></Variable>
                                </Assign>
                            </Assignment>
                            <Call value="FMT001A0"></Call>
                        </Statements>
                    </Action>
                    <Action>
                        <Name value="_start_"></Name>
                        <Statements>
                            <Call value="Z"></Call>
                        </Statements>
                    </Action>
                    <Action>
                        <Name value="FMT001A0"></Name>
                        <Statements>
                            <Comment value="*************************************************************************"></Comment>
                            <Comment value="*    FMT001A0    TEST PROGRAM - CONTROL MODULE                                  *"></Comment>
                            <Comment value="*************************************************************************"></Comment>
                            <Comment value="*"></Comment>
                            <Comment value="*"></Comment>
                            <Comment value="*              PRINT NOGEN"></Comment>
                            <Call value="A_000000"></Call>
                        </Statements>
                    </Action>
                    <Action>
                        <Name value="A_000000"></Name>
                        <Statements>
                            <Comment value="&lt;FermaT&gt; 00000035 "></Comment>
                            <A_Proc_Call>
                                <Name value="push_regs"></Name>
                                <Expressions>
                                    <Variable value="r0"></Variable>
                                    <Variable value="r1"></Variable>
```

```
                    <Variable value="r2"></Variable>
                    <Variable value="r3"></Variable>
                    <Variable value="r4"></Variable>
                    <Variable value="r5"></Variable>
                    <Variable value="r6"></Variable>
                    <Variable value="r7"></Variable>
                    <Variable value="r8"></Variable>
                    <Variable value="r9"></Variable>
                    <Variable value="r10"></Variable>
                    <Variable value="r11"></Variable>
                    <Variable value="r12"></Variable>
                    <Variable value="r13"></Variable>
                    <Variable value="r14"></Variable>
                </Expressions>
                <Lvalues>
                    <Var_Lvalue value="reg_stack"></Var_Lvalue>
                </Lvalues>
            </A_Proc_Call>
            <Call value="A_000004"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000004"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000036 "></Comment>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r3"></Var_Lvalue>
                    <Variable value="r15"></Variable>
                </Assign>
            </Assignment>
            <Call value="A_000006"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000006"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000038 WSAVE WSAVE[5] WSAVE[6] WSAVE[7] WSAVE[8]"></Comment>
            <Assignment>
                <Assign>
                    <Sub_Seg_Lvalue>
                        <Var_Lvalue value="WSAVE"></Var_Lvalue>
                        <Number value="5"></Number>
                        <Number value="8"></Number>
                    </Sub_Seg_Lvalue>
                    <Variable value="r13"></Variable>
                </Assign>
            </Assignment>
            <Call value="A_00000A"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_00000A"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000039 WSAVE"></Comment>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r14"></Var_Lvalue>
                    <X_Funct_Call>
                        <Name value="address_of"></Name>
                        <Expressions>
                            <Variable value="WSAVE"></Variable>
                        </Expressions>
                    </X_Funct_Call>
                </Assign>
            </Assignment>
            <Call value="A_00000E"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_00000E"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000040 "></Comment>
            <Assignment>
                <Assign>
                    <Rel_Seg_Lvalue>
                        <Var_Lvalue value="a"></Var_Lvalue>
                        <Plus>
                            <Variable value="r13"></Variable>
                            <Number value="8"></Number>
                        </Plus>
                        <Number value="4"></Number>
                    </Rel_Seg_Lvalue>
                    <Variable value="r14"></Variable>
                </Assign>
            </Assignment>
            <Call value="A_000012"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000012"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000041 WSAVE"></Comment>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r13"></Var_Lvalue>
                    <X_Funct_Call>
                        <Name value="address_of"></Name>
```

```
                        <Expressions>
                            <Variable value="WSAVE"></Variable>
                        </Expressions>
                    </X_Funct_Call>
                </Assign>
            </Assignment>
            <Call value="A_000016"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000016"></Name>
        <Statements>
            <Comment value="*"></Comment>
            <Comment value="&amp;lt;FermaT&amp;gt; 00000043 WTOTAL"></Comment>
            <A_Proc_Call>
                <Name value="zap"></Name>
                <Expressions>
                    <X_Funct_Call>
                        <Name value="p_lit"></Name>
                        <Expressions>
                            <Number value="1"></Number>
                            <Number value="1"></Number>
                            <String value="0"></String>
                        </Expressions>
                    </X_Funct_Call>
                </Expressions>
                <Lvalues>
                    <Var_Lvalue value="WTOTAL"></Var_Lvalue>
                </Lvalues>
            </A_Proc_Call>
            <Cond>
                <Guarded>
                    <X_BFunct_Call>
                        <Name value="dec_eq"></Name>
                        <Expressions>
                            <Variable value="WTOTAL"></Variable>
                            <X_Funct_Call>
                                <Name value="p_lit"></Name>
                                <Expressions>
                                    <Number value="1"></Number>
                                    <Number value="4"></Number>
                                    <String value="0"></String>
                                </Expressions>
                            </X_Funct_Call>
                        </Expressions>
                    </X_BFunct_Call>
                    <Statements>
                        <Assignment>
                            <Assign>
                                <Var_Lvalue value="cc"></Var_Lvalue>
                                <Number value="0"></Number>
                            </Assign>
                        </Assignment>
                    </Statements>
                </Guarded>
                <Guarded>
                    <X_BFunct_Call>
                        <Name value="dec_less"></Name>
                        <Expressions>
                            <Variable value="WTOTAL"></Variable>
                            <X_Funct_Call>
                                <Name value="p_lit"></Name>
                                <Expressions>
                                    <Number value="1"></Number>
                                    <Number value="4"></Number>
                                    <String value="0"></String>
                                </Expressions>
                            </X_Funct_Call>
                        </Expressions>
                    </X_BFunct_Call>
                    <Statements>
                        <Assignment>
                            <Assign>
                                <Var_Lvalue value="cc"></Var_Lvalue>
                                <Number value="1"></Number>
                            </Assign>
                        </Assignment>
                    </Statements>
                </Guarded>
                <Guarded>
                    <True></True>
                    <Statements>
                        <Assignment>
                            <Assign>
                                <Var_Lvalue value="cc"></Var_Lvalue>
                                <Number value="2"></Number>
                            </Assign>
                        </Assignment>
                    </Statements>
                </Guarded>
            </Cond>
            <Call value="A_00001C"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_00001C"></Name>
        <Statements>
            <Comment value="&amp;lt;FermaT&amp;gt; 00000044 WCT"></Comment>
```

```
                <A_Proc_Call>
                    <Name value="ap"></Name>
                    <Expressions>
                        <X_Funct_Call>
                            <Name value="p_lit"></Name>
                            <Expressions>
                                <Number value="1"></Number>
                                <Number value="1"></Number>
                                <String value="1"></String>
                            </Expressions>
                        </X_Funct_Call>
                    </Expressions>
                    <Lvalues>
                        <Var_Lvalue value="WCT"></Var_Lvalue>
                    </Lvalues>
                </A_Proc_Call>
                <Cond>
                    <Guarded>
                        <X_BFunct_Call>
                            <Name value="dec_eq"></Name>
                            <Expressions>
                                <Variable value="WCT"></Variable>
                                <X_Funct_Call>
                                    <Name value="p_lit"></Name>
                                    <Expressions>
                                        <Number value="1"></Number>
                                        <Number value="4"></Number>
                                        <String value="0"></String>
                                    </Expressions>
                                </X_Funct_Call>
                            </Expressions>
                        </X_BFunct_Call>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="0"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                    <Guarded>
                        <X_BFunct_Call>
                            <Name value="dec_less"></Name>
                            <Expressions>
                                <Variable value="WCT"></Variable>
                                <X_Funct_Call>
                                    <Name value="p_lit"></Name>
                                    <Expressions>
                                        <Number value="1"></Number>
                                        <Number value="4"></Number>
                                        <String value="0"></String>
                                    </Expressions>
                                </X_Funct_Call>
                            </Expressions>
                        </X_BFunct_Call>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="1"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                    <Guarded>
                        <True></True>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="2"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                </Cond>
                <Call value="A_000022"></Call>
            </Statements>
        </Action>
        <Action>
            <Name value="A_000022"></Name>
            <Statements>
                <Comment value="&amp;lt;FermaT&amp;gt; 00000045 WNUM"></Comment>
                <A_Proc_Call>
                    <Name value="ap"></Name>
                    <Expressions>
                        <X_Funct_Call>
                            <Name value="p_lit"></Name>
                            <Expressions>
                                <Number value="1"></Number>
                                <Number value="1"></Number>
                                <String value="1"></String>
                            </Expressions>
                        </X_Funct_Call>
                    </Expressions>
                    <Lvalues>
                        <Var_Lvalue value="WNUM"></Var_Lvalue>
```

```
                    </Lvalues>
                </A_Proc_Call>
                <Cond>
                    <Guarded>
                        <X_BFunct_Call>
                            <Name value="dec_eq"></Name>
                            <Expressions>
                                <Variable value="WNUM"></Variable>
                                <X_Funct_Call>
                                    <Name value="p_lit"></Name>
                                    <Expressions>
                                        <Number value="1"></Number>
                                        <Number value="4"></Number>
                                        <String value="0"></String>
                                    </Expressions>
                                </X_Funct_Call>
                            </Expressions>
                        </X_BFunct_Call>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="0"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                    <Guarded>
                        <X_BFunct_Call>
                            <Name value="dec_less"></Name>
                            <Expressions>
                                <Variable value="WNUM"></Variable>
                                <X_Funct_Call>
                                    <Name value="p_lit"></Name>
                                    <Expressions>
                                        <Number value="1"></Number>
                                        <Number value="4"></Number>
                                        <String value="0"></String>
                                    </Expressions>
                                </X_Funct_Call>
                            </Expressions>
                        </X_BFunct_Call>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="1"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                    <Guarded>
                        <True></True>
                        <Statements>
                            <Assignment>
                                <Assign>
                                    <Var_Lvalue value="cc"></Var_Lvalue>
                                    <Number value="2"></Number>
                                </Assign>
                            </Assignment>
                        </Statements>
                    </Guarded>
                </Cond>
                <Call value="A_000028"></Call>
            </Statements>
        </Action>
        <Action>
            <Name value="A_000028"></Name>
            <Statements>
                <Comment value="*"></Comment>
                <Comment value="&lt;FermaT&gt; 00000047 "></Comment>
                <A_Proc_Call>
                    <Name value="FMT001A1"></Name>
                    <Expressions>
                        <String value="FMT001A1"></String>
                    </Expressions>
                    <Lvalues>
                        <Var_Lvalue value="result_code"></Var_Lvalue>
                        <Var_Lvalue value="os"></Var_Lvalue>
                    </Lvalues>
                </A_Proc_Call>
                <Assignment>
                    <Assign>
                        <Var_Lvalue value="r15"></Var_Lvalue>
                        <Variable value="result_code"></Variable>
                    </Assign>
                </Assignment>
                <Call value="A_000036"></Call>
            </Statements>
        </Action>
        <Action>
            <Name value="A_000036"></Name>
            <Statements>
                <Comment value="&lt;FermaT&gt; 00000054 "></Comment>
                <A_Proc_Call>
                    <Name value="FMT001A2"></Name>
                    <Expressions>
                        <String value="FMT001A2"></String>
```

```
                </Expressions>
                <Lvalues>
                    <Var_Lvalue value="result_code"></Var_Lvalue>
                    <Var_Lvalue value="os"></Var_Lvalue>
                </Lvalues>
            </A_Proc_Call>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r15"></Var_Lvalue>
                    <Variable value="result_code"></Variable>
                </Assign>
            </Assignment>
            <Call value="A_000046"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000046"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000061 "></Comment>
            <A_Proc_Call>
                <Name value="FMT001A3"></Name>
                <Expressions>
                    <String value="FMT001A3"></String>
                </Expressions>
                <Lvalues>
                    <Var_Lvalue value="result_code"></Var_Lvalue>
                    <Var_Lvalue value="os"></Var_Lvalue>
                </Lvalues>
            </A_Proc_Call>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r15"></Var_Lvalue>
                    <Variable value="result_code"></Variable>
                </Assign>
            </Assignment>
            <Call value="A_000056"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_000056"></Name>
        <Statements>
            <Comment value="*"></Comment>
            <Comment value="&lt;FermaT&gt; 00000072 WSAVE WSAVE[5] WSAVE[6] WSAVE[7] WSAVE[8]"></Comment>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r13"></Var_Lvalue>
                    <Sub_Seg>
                        <Variable value="WSAVE"></Variable>
                        <Number value="5"></Number>
                        <Number value="8"></Number>
                    </Sub_Seg>
                </Assign>
            </Assignment>
            <Call value="A_00005A"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_00005A"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000073 "></Comment>
            <A_Proc_Call>
                <Name value="pop_regs"></Name>
                <Expressions></Expressions>
                <Lvalues>
                    <Var_Lvalue value="r0"></Var_Lvalue>
                    <Var_Lvalue value="r1"></Var_Lvalue>
                    <Var_Lvalue value="r2"></Var_Lvalue>
                    <Var_Lvalue value="r3"></Var_Lvalue>
                    <Var_Lvalue value="r4"></Var_Lvalue>
                    <Var_Lvalue value="r5"></Var_Lvalue>
                    <Var_Lvalue value="r6"></Var_Lvalue>
                    <Var_Lvalue value="r7"></Var_Lvalue>
                    <Var_Lvalue value="r8"></Var_Lvalue>
                    <Var_Lvalue value="r9"></Var_Lvalue>
                    <Var_Lvalue value="r10"></Var_Lvalue>
                    <Var_Lvalue value="r11"></Var_Lvalue>
                    <Var_Lvalue value="r12"></Var_Lvalue>
                    <Var_Lvalue value="r13"></Var_Lvalue>
                    <Var_Lvalue value="r14"></Var_Lvalue>
                    <Var_Lvalue value="reg_stack"></Var_Lvalue>
                </Lvalues>
            </A_Proc_Call>
            <Call value="A_00005E"></Call>
        </Statements>
    </Action>
    <Action>
        <Name value="A_00005E"></Name>
        <Statements>
            <Comment value="&lt;FermaT&gt; 00000074 "></Comment>
            <Assignment>
                <Assign>
                    <Var_Lvalue value="r15"></Var_Lvalue>
                    <Number value="0"></Number>
                </Assign>
            </Assignment>
            <Cond>
                <Guarded>
                    <Equal>
```

```
                                <Variable value="r15"></Variable>
                                <Number value="0"></Number>
                            </Equal>
                            <Statements>
                                <Assignment>
                                    <Assign>
                                        <Var_Lvalue value="cc"></Var_Lvalue>
                                        <Number value="0"></Number>
                                    </Assign>
                                </Assignment>
                            </Statements>
                        </Guarded>
                        <Guarded>
                            <True></True>
                            <Statements>
                                <Assignment>
                                    <Assign>
                                        <Var_Lvalue value="cc"></Var_Lvalue>
                                        <Number value="2"></Number>
                                    </Assign>
                                </Assignment>
                            </Statements>
                        </Guarded>
                    </Cond>
                    <Call value="A_000060"></Call>
                </Statements>
            </Action>
            <Action>
                <Name value="A_000060"></Name>
                <Statements>
                    <Comment value="&amp;lt;FermaT&amp;gt; 00000075 "></Comment>
                    <Assignment>
                        <Assign>
                            <Var_Lvalue value="destination"></Var_Lvalue>
                            <Variable value="r14"></Variable>
                        </Assign>
                    </Assignment>
                    <Call value="dispatch"></Call>
                </Statements>
            </Action>
            <Action>
                <Name value="dispatch"></Name>
                <Statements>
                    <Cond>
                        <Guarded>
                            <Equal>
                                <Variable value="destination"></Variable>
                                <Number value="0"></Number>
                            </Equal>
                            <Statements>
                                <Call value="Z"></Call>
                            </Statements>
                        </Guarded>
                        <Guarded>
                            <True></True>
                            <Statements>
                                <Comment value="Unknown destination "></Comment>
                                <Call value="Z"></Call>
                            </Statements>
                        </Guarded>
                    </Cond>
                </Statements>
            </Action>
        </Actions>
    </A_S>
  </Statements>
</Var>

 </Statements>
```

Appendix C: List of Publications

[1] Ruimin Liu and Hongji Yang, "Software Evolution in Ubiquitous Computing", *Proceedings of CACSUK 2003*, Manchester, England, September, 2003.

[2] Ruimin Liu and Yunsheng Zhang, "Information Integration in Terminal Point Control of Copper Smelt Process", *Proceedings of CACSUK 2004*, Liverpool, UK, 2004.

[3] Ruimin Liu, Hongji Yang, Yangsheng Wang and Wei Pang, "An Evolutionary System Development Approach in a Pervasive Computing Environment", *IEEE International Conference on Cyberworlds (IEEE CW 2004)*, November 2004.

[4] Ruimin Liu, Feng Chen and Hongji Yang, "Agent-based Web Services Evolution for Pervasive Computing", *The 11th Asia Pacific Software Engineering Conference, EUSE Workshop*, September 2004.

[5] Zhuopeng Zhang, Ruimin Liu and Hongji Yang, "Service Identification and Packaging in Service Oriented Reengineering", *The 17th International Conference on Software Engineering and Knowledge Engineering (SEKE2005)*, July 2005.

[6] Feng Chen, Hong Zhou, Ruimin Liu and Hongji Yang, "An Ontology-Based Approach to Portable Embedded System Development", *The International Conference on Software Engineering and Knowledge Engineering (SEKE2009)*, July 2009.